國　立　交　通　大　學

運輸科技與管理學系碩士班

碩士論文

應用變數產生法求解有時間窗限制的收送貨問題

A Column Generation Approach for the Pickup and Delivery Problem with Time Windows

研究生：葉珮婷

指導教授：王晉元

中　華　民　國　九　十　八　年　七　月

應用變數產生法求解有時間窗限制的收送貨問題
A Column Generation Approach for the Pickup and Delivery Problem with Time Windows

研 究 生：葉珮婷　　Student：Pei-Ting Yeh

指導教授：王晉元　　Advisor：Jin-Yuan Wang

國 立 交 通 大 學
運 輸 科 技 與 管 理 學系
碩 士 論 文

A Thesis
Submitted to Department of Transportation Technology and Management
College of Management
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Transportation Technology and Management

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

應用變數產生法求解有時間窗限制的收送貨問題

學生：葉珮婷　　指導教授：王晉元

國立交通大學運輸科技與管理學系碩士班

摘　要

本研究目的為替貨運業之預先派遣作業產生具有效率的路線，除了須考量路徑成本外，尚需考量貨運業收送貨之特性，包含優先限制、聯結限制、時間窗限制與車容量限制。本研究將有時間窗限制的收送貨問題(Pickup and Delivery Problem with Time Windows, PDPTW)建構為一集合分割問題，並提出兩套以變數產生法(Column Generation)為基礎的演算法求解。此兩套演算法皆以變數產生法、分支定限法與解決重覆涵蓋問題之啟發式解法三大部分所組成，兩套演算法不同之處在於變數產生法之求解子問題演算法。本研究將子問題設計為考慮有時間窗限制收送貨特性之最短路徑問題，第一套演算法之子問題使用修正後的 Dijkstra's 演算法與啟發式解法求解；第二套演算法之子問題使用修正後之 Label Correcting 演算法求解。

測試結果發現第一套演算法與第二套演算法於小例題測試範例之求解績效誤差在5%內，但使用第一套演算法之求解時間遠較第二套演算法快。本研究將第一套演算法求解子問題時之啟發式演算法分為三種組合方式，測試結果發現三種組合方式對於平均旅行成本在小範圍時間窗客戶群與同一客戶大小有顯著差異；三種組合方式對於平均車輛數僅在 LR2 客戶類型無顯著差異，在其餘客戶類型與同一客戶大小皆有顯著差異；三種組合方式對於不包含分支定限法之平均求解時間皆為第三種組合最快。由測試結果可知使用第一套演算法且子問題啟發式演算法使用第二種組合方式能在較快速之運算時間內求得不錯的路徑組合。

關鍵字：有時間窗限制的收送貨問題，集合分割問題，變數產生法

A Column Generation Approach for the Pickup and Delivery Problem with Time Windows

Student：Pei-Ting Yeh　　Advisor：Dr. Jin-Yuan Wang

Department of Transportation Technology and Management
National Chiao Tung University

ABSTRACT

Pickup and delivery problem with time windows (PDPTW) is an important and fundamental problem for many logistic service providers. We first formulate this problem as a set-partitioning problem. Two column generation based algorithms are proposed for solving this model. Both algorithms consist of three phases: column generation, branch-and-bound, and heuristic local search. The column generation subproblem is formulated as a shortest path problem with multiple side constraints. The major differences between these two algorithms are the adopted solution techniques. The first algorithm is based on the traditional Dijkstra's algorithm and the second algorithm is basically a modified Label Correcting.

We use benchmark testing problems to evaluate the performance of these two algorithms. Numerical experiments show that the proposed algorithms can find near optimal solution for smaller test problems, and the second algorithm could get better solutions efficiently on large test problems generalized from benchmark problems in the literature.

Keywords: PDPTW, set-partitioning problem, Column Generation

誌謝

感謝交通大學運輸科技與管理學系的老師們，讓我在交大運管系的六年間獲得了不少知識，讓我於六年間成長不少。感謝指導教授王晉元老師兩年間的用心指導，每星期抽空費心指導，讓我每個星期能夠不斷的督促自己努力向上，能夠有這本論文的產出，要歸功於老師像燈塔般，引領著我一步步的領略論文的精髓，使我可以很紮實的將每一個步驟完成。

感謝論文期中報告審查委員韓復華老師與黃寬丞老師，於論文期中報告給予之寶貴建議與評論，讓我對論文期中報告後續研究更加謹慎小心；感謝論文口試委員中華大學卓裕仁老師與蘇昭銘老師，於口試前撥冗審閱此論文，並給予許多寶貴建議，使本論文能夠更加完善，在此致上十二萬分的謝意。

感謝實驗室的學長們、同學們與學弟妹們，讓我研究所兩年中有歡笑與扶持，其中最要感謝的是程式能力很強的學妹家儒，一起在 CVO 計畫中打拼，雖然於論文程式中無得到你的幫助，但常於凌晨離開學校前讓我發洩情緒，有個抒發情緒的管道。也要感謝韓老師實驗室的俊德學長，總是不厭其煩的讓我使用電腦跑程式，也提供我論文上的建議，使本論文結果得以順利完成，我永銘記在心。

最重要的是感謝我的雙親，25 年來讓我衣食無缺，給予無限的愛與關懷，使我無後顧之憂的求學，若沒有您們從小的教養，就沒有現在的我。在撰寫論文程式的過程中，因常凌晨始離開學校，新竹治安不佳，導致常使您們擔心，讓我感到抱歉萬分，但也因您們的體諒與關心，使我將此篇論文順利完成；感謝阿公阿嬤於我成長的過程中教導我許多事情，論文撰寫過程中較晚回家時阿公的噓寒問暖，總是讓我感受到無比的幸福；感謝當老師的姊姊在我成長的過程中永遠扮演著垃圾桶的角色，在我抱怨與不愉快需要發洩時，總是靜靜的聽我說話與適時的給我建議，並不厭其煩的發揮國文能力，幫助我於論文文字上的校正與美化；感謝比我高大不擅言詞的弟弟於心靈上的支持，並適時的發揮馬殺雞的能力，幫我解除肌肉疲勞，讓我快速的恢復體力，繼續撰寫論文。還有感謝永遠的班長邦展，生活上讓我有傾訴的對象，遇到論文程式撰寫的瓶頸也讓我有所依靠。

最後，誠心的感謝我認識的所有人，並與您們分享完成論文的喜悅與成就感。

葉珮婷 謹誌

中華民國九十八年七月 于風城

目錄

表目錄

圖目錄

第一章　序論

1.1 研究動機

近年來全球商業的競爭趨勢改變，從企業與企業的競爭逐漸擴大至供應鏈與供應鏈間的競爭。供應鏈內的企業組織均需配合整個供應鏈中的流程與活動，以降低成本，提升效率與品質，並共享其利潤。物流配送作業為供應鏈中相當重要的一個環節，佔有不少的成本，因此受到全球大小企業的重視與關切。在物流配送作業中，車輛的利用與規劃的配送路線佔物流企業營運相當大的成本。因此在物流配送的相關領域中，車輛途程問題(Vehicle Routing Problem, VRP)在國內外已有相當廣泛的研究[6][24]。

現今提供物流服務的汽車貨運業中，其一型態為點對點運輸，顧客提供物流資訊要求貨運公司派車至指定點收貨(Pickup)再送至另一指定點(Delivery)，此類問題為收送貨問題(Pickup and Delivery Problem, PDP)，為VRP的延伸，VRP為替車輛規劃巡迴路徑的問題，在這類的問題中每輛車均由場站出發，滿足所有顧客需求點，顧客的需求只有單純的收貨或是送貨的服務，且考慮在每個顧客點只能服務一次的限制下，車輛最終回到場站。而收送貨問題(PDP)所服務的顧客必定同時包含一收貨作業與一送貨作業，且分別有不同之作業地點，必須以同一車輛進行服務，且須先完成收貨作業才可進行送貨作業。由於JIT(Just in Time)概念興起，每個顧客需求點有時間窗限制，業者在消費意識高漲與競爭壓力的促使下，必須提供快速、穩定、正確、準時的服務。因此具有時間窗限制的收送貨問題(Pickup and Delivery Problem with Time Windows, PDPTW)的研究將益發重要。

實務上，貨車的調度派遣作業因考量的情況複雜多變，使得調度派遣作業通常缺乏系統化的支援，僅依據派遣人員以往的經驗來指派任務與決定貨車行走的路線，因此容易作出多繞路的路徑組合，造成公司人力與資源的浪費，長期如此必對公司的獲利能力造成影響。

貨運業的調度派遣作業可分為預先派遣和即時派遣。預先派遣為訂單於實際需求前幾天即下單，到了實際需求前一天一起將隔天需派遣之訂單實行派遣作業。即時派遣則為當天接到訂單，馬上調度派遣。因此若使用系統化派遣時，即時派遣需求之系統需能夠快速的求解，相對於預先派遣來說，預先派遣只需在前一天求出解即可。

PDPTW已知屬於NP-hard問題[24]，運算時間會隨著問題規模增加而呈指數性的成長，無法在合理的時間內求得最佳解，因此近年來許多屬於NP-hard問題皆朝向巨集啟發式演算法(Meta-Heuristics) 的方向發展，啟發式演算法雖可較快速的求解，可用於實務上的即時派遣，但容易陷入區域最佳解，且其所得出的解較差。

本研究針對貨運業之預先派遣作業提供車輛路徑組合，因此不需為了快速求解，退而求其次使用啟發式解法求得較差的解。本研究將針對PDPTW問題建構一數學模式，

以最佳解為基礎的演算法求解該數學模式，替調度人員產生一組同時滿足顧客需求與貨運限制的車輛路徑組合。

1.2 研究目的

本研究針對貨運業預先派遣與調度的情形，發展出一收送貨路徑規劃方法，使車輛路徑的規劃作業自動化，以減少人工調度負擔，並降低貨運業與人工派遣與調度時所帶來的損失，提升貨物運送效率與降低貨物運送的成本。同時，發展一套適當的求解演算方法，進行相關的範例驗證。

1.3 研究範圍

本小節首先定義 PDPTW 問題相關名詞，再說明本研究所探討的 PDPTW 問題，PDPTW 問題相關名詞定義如下：

1. 任務：貨運公司接獲顧客的訂單(貨運需求)後，即產生一個收貨任務與一個送貨任務的配對。貨運公司需在指定的時間窗內派車前往收貨點載貨，之後在指定的時間窗內前往送貨點執行送貨任務，完成服務此訂單的收送貨後才算完成此筆貨運需求。在路網中，一收貨任務代表一收貨點，一送貨任務代表一送貨點。
2. 巡迴路徑：每輛貨車從場站出發後執行一連串的任務，最終再回到場站的路徑稱為巡迴路徑，簡稱路徑。如圖 1 所示貨車從場站出發後行走的巡迴路徑為：場站-收貨任務 1-送貨任務 1-收貨任務 2-送貨任務 2-場站。
3. 優先限制(Precedence Constraints)：針對每一貨運需求，收貨任務的服務順序必須於送貨任務前。
4. 聯結限制(Paring Constraints) ：每一貨運需求之收送貨任務均由同一輛車服務。

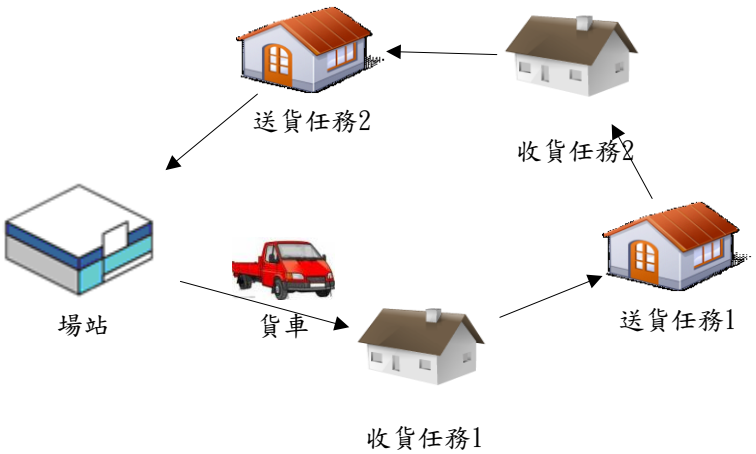

圖 1 巡迴路徑示意圖

本研究所探討的具時間窗限制的收送貨問題(PDPTW)為多車輛、單車種、單場站，每輛車載重容量相同，且車輛數有上限。貨物需求點已知，每個貨運需求即代表一組收貨與送貨的配對，故每個收貨任務必有一送貨任務與其配對，每個送貨任務亦必有一收貨任務與其配對。每個收送貨任務均有允許服務之時間範圍，車輛僅能於該任務之時間範圍內執行任務，若於時間範圍前到達，則需等待至最早允許執行時間，產生等待時間，並假設服務時間為0，即車輛開始執行服務後即完成任務離開。每個任務皆有貨物載重量；若為收貨任務則貨物載重量為正值，若為送貨任務則貨物載重量為負值。每輛車從場站出發，最後回到場站。每輛車均為混合式收送貨，亦即在車輛離開場站至回到場站之路徑中，允許執行不同訂單之收貨或送或任務，但同一訂單之收貨任務必在送貨任務前執行，且車輛一離開場站須直接執行收貨任務。本研究PDPTW問題為求解總旅行距離成本最小的車輛巡迴路徑組合。本研究探討之PDPTW問題特性如表1所示：

表 1 本研究之 PDPTW 問題

項目	說明
目標	最小化總旅行距離
設施資源	單場站 單車種，多車輛 車輛有容量限制
顧客需求	需求點已知且不變 包含一收貨點作業與一送貨點作業(聯結限制) 各作業點有時間窗限制
節點服務	混合收送貨 同一組的收送貨點，必先收貨後送貨(優先限制)

1.4 研究流程

本研究流程如圖2所示，首先界定問題，繼而回顧PDPTW相關文獻與求解方法之相關研究，包括啟發式解法、最佳解解法等，分析各式解法之優缺點與適用性，挑選適合本研究之求解方法。接著，建構一能夠滿足本研究之收送貨問題的數學模式，並設計一套演算法求解該模式。之後，本研究進行範例測試與分析。最後根據分析結果，提出結論與建議。

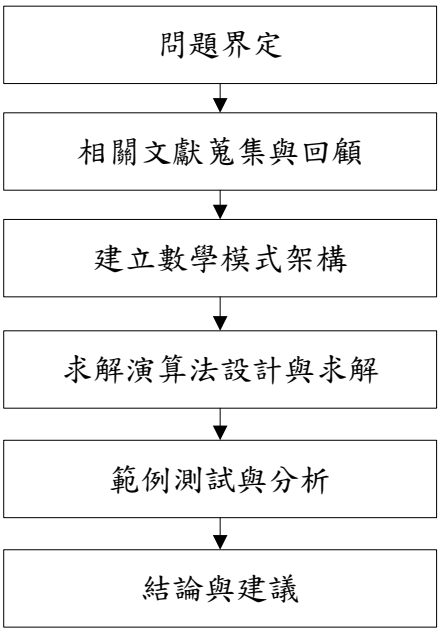

圖 2 研究流程圖

第二章　文獻回顧

本研究先介紹一般化收送貨問題(The General Pickup and Delivery Problem, GPDP)，並探討過去文獻針對PDP與PDPTW定義與數學規劃模式中限制式間之差異，以及各式演算法求解方式，包含啟發式演算法與最佳解解法。2.1節介紹一般化收送貨問題；2.2節探討PDP與PDPTW定義與數學規劃模式中限制式間之差異；2.3節回顧PDPTW之各式求解演算法；2.4節文獻回顧小結。

2.1 一般化收送貨問題簡介

一般化收送貨問題在於替一貨車車隊做路徑規劃，顧客需求包含收貨作業與送貨作業兩部分，車輛在不違反容量限制下，服務一連串的收貨點與送貨點。收送貨問題可依配送方式不同分為兩大類[3][17]：車輛載貨從場站出發將貨物運送給送貨點顧客，再至收貨點顧客收取貨物運回場站，此種將貨物從場站至場站的配送方式為「回程取貨車輛途程問題 (Vehicle Routing Problems with Backhauls , VRPB)」。若車輛從場站出發至顧客指定地點收取貨物，將貨物送到顧客指定地點送貨，此種將貨物從顧客點直接配送至另一顧客點之配送方式為收送貨車輛途程問題（Vehicle Routing Problems with Pickups and Deliveries, VRPPD），如圖3所示。

VRPPD可依其收送貨點是否配對再細區分。若運送皆為相同商品，從一收貨點取貨後，可送至任一送貨點，此為收送貨點無配對之收送貨問題(Pickup and Delivery Vehicle Routing Problem, PDVRP)。另收送貨點有配對之收送貨問題，若運送為人則分類為撥召問題 (Dial-A-Ride Problem, DARP)，若運送為物品則分類為有配對的收送貨問題簡稱收送貨問題(Pickup and Delivery Problem, PDP)。

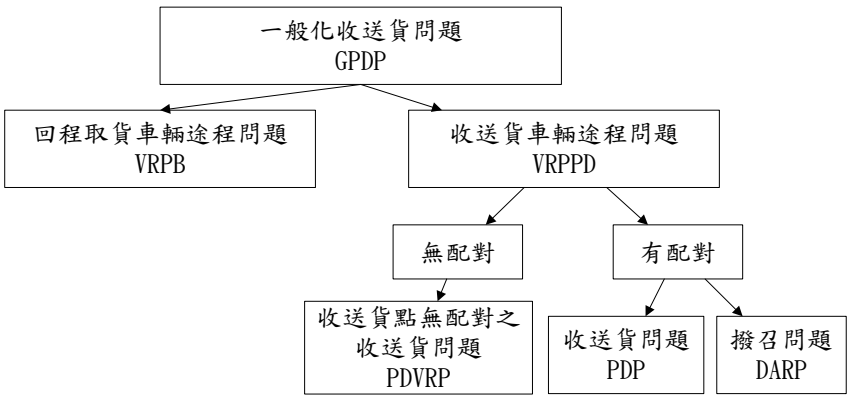

圖3一般化收送貨問題分類

資料來源：[16][17]

2.2 PDP 與 PDPTW 問題簡介

PDP是VRP的延伸問題，顧客需求(訂單)為一組收送貨作業，收貨與送貨作業可交錯服務，但同一組的收貨作業(Pickup)需在送貨作業(delivery)前完成，即為優先限制(precedence)；且每個貨運需求的收送貨作業均由同一輛車執行，即為聯結限制(pairing)。PDPTW與PDP間之差異為PDPTW中之各作業點存在服務時間限制[21]。

2.2.1 PDP 與 PDPTW 定義

PDP定義與VRP相似，但PDP之顧客需求為一組收送貨作業，不再僅是收貨或送貨，且同一組的收貨作業需於送貨作業前完成服務，並由同一輛車服務。因此PDP問題可描述為：客戶需求包含收貨作業與送貨作業兩部分，車輛由場站空車出發到顧客指定的收貨地點收取貨物運送至顧客指定的送貨地點，任務完成後回到場站，配送過程不違反車輛容量限制；目標為最小化車輛數及車輛路線的總運輸成本。

PDP問題定義如下：

1. 各車輛的起迄點須為同一場站。
2. 每個顧客需求包含一個收貨作業與一個送貨作業，須由同一車輛進行服務，且只能服務一次。
3. 一顧客需求指派給一車輛進行服務，則車輛的行駛路線必須包含該顧客需求的收貨作業與送貨作業，且收貨作業的服務順序必須於送貨作業前。
4. 各車輛的載貨量無論何時皆不能超過車容量上限。
5. 車輛於作業點完成後立即離開，並前往下一個作業點。

PDPTW 與PDP 差異僅在於各項需求作業新增了服務時間窗的限制，因此問題定義較PDP新增了下列3 項。

6. 到達作業點 i 的時間 (A_i) 若早於作業點 i 的最早開始服務時間 (e_i)，則必須等待到 ei 時間點時，才能開始服務。
7. 各作業點 i 需停留一段時間 (s_i) 進行服務，待服務完成後方能離開。
8. 離開作業點 i 的時間 (Di)，若晚於作業點 i 的最晚開始執行時間 (l_i)，則出現延遲狀況。

2.2.2 數學規劃模式

求解PDPTW相關問題時，大多數研究皆同時考量最小化車輛數與路徑成本[7][24]，亦有少許研究針對問題不同之特性，加入最小化等待時間 (車輛閒置時間)、最小化服務延遲時間或最小化顧客不便等其他次要目標，成為多目標優化問題[13][25]。文獻上已有許多學者提出 PDPTW 的數學規劃模式[2][5][10][19]。

PDP中參數與集合定義如下：

n: 訂單數

0：起點

$2n$+1：迄點

其中起迄點為同一場站，僅車輛流入與流出之不同差異。

N_i^+： 訂單 i 之收貨作業的地點 ，$i \in N^+$

N^+：收貨點作業的集合，$N^+ := \cup_{i \in N} N_i^+$ $N^+ = \{1, \cdots, n\}$

N_i^-：訂單 i 之送貨作業的地點，$i \in N^-$

N^-：所有送貨點的集合，$N^- := \cup_{i \in N} N_i^-$ $N^- = \{n+1, \cdots, 2n\}$

N ：所有作業點的集合，$N := N^+ \cup N_i^-$

q_i：作業點 i 的貨物量。若$i \in N^+$，q_i為正數；若$i \in N^-$，q_i為負數；若$i = 0$ 或 $i = 2n + 1$，$q_i = 0$

K：所有車輛之集合

Q_k：車輛 k 的容量上限

c_{ij}^k：$k \in K$，$i, j \in V$，$i \neq j$車輛 k 從作業點 i 到 j 的旅行成本

t_{ij}^k：$k \in K$，$i, j \in V$，$i \neq j$車輛 k 從作業點 i 到 j 的旅行時間

s_i：作業點 i 所需的服務時間

V：所有節點的集合， $V = \{0, 2n + 1\} \cup N^+ \cup N^-$

A：所有節線的集合， $A = \{(i, j): i, j \in V, i \neq n + 2n + 1, j \neq 0, i \neq j\}$

PDP 數學規劃模式中的決策變數如下：

x_{ij}^k：$k \in K$，$i,j \in V$，$i \neq j$，若車輛從作業點 i 到作業點 j，$x_{ij}^k = 1$；否則$x_{ij}^k = 0$

L_i^k：車輛 k 離開節點 i 時累積的承載量

T_i^k：車輛 k 開始服務作業點 i 的時間

PDPTW 中增加的參數如下：

$[e_i, l_i]$：作業點 i 的時間窗限制。e_i為最早可開始服務時間。l_i為最晚可開始服務時間。

PDP 與 PDPTW 之定式以[16]所提出之模式為例，PDP 之定式如式(1)~(11)所示。

$$Minimize \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \quad (1)$$

Subject to

$$\sum_{k \in K} \sum_{j:(i,j) \in A} x_{ij}^k = 1 \qquad \forall i \in N, \quad (2)$$

$$\sum_{j:(0,j) \in A} x_{0j}^k = 1 \qquad \forall k \in K, \quad (3)$$

$$\sum_{i:(i,2n+1) \in A} x_{i,2n+1}^k = 1 \qquad \forall k \in K, \quad (4)$$

$$\sum_{i:(i,j) \in A} x_{ij}^k - \sum_{i:(j,i) \in A} x_{ji}^k = 0 \qquad \forall k \in K,\ j \in N, \quad (5)$$

$$\sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(n+i,j) \in A} x_{n+i,j}^k = 0 \qquad \forall k \in K,\ i \in N, \quad (6)$$

$$x_{ij}^k \left(L_i^k + q_j - L_j^k \right) = 0 \qquad \forall k \in K,\ (i,j) \in A \quad (7)$$

$$T_i^k \leq T_{n+i}^k \qquad \forall i \in N^+, k \in K \quad (8)$$

$$x_{ij}^k \left(T_i^k + s_i + t_{ij}^k - T_j^k \right) = 0 \qquad \forall k \in K,\ (i,j) \in A \quad (9)$$

$$\max\{0, q_i\} \leq L_i^k \leq \min\left\{ Q^k, Q^k + q_i \right\} \qquad \forall k \in K,\ i \in V, \quad (10)$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K, (i,j) \in A \quad (11)$$

上述數學規劃式中，目標函數(1)為最小化總旅行成本；限制式(2)限制每一顧客需求(包含一收貨作業和一送貨作業)被同一輛車服務，且恰好一次；限制式(3)(4)表每輛車從場站出發，最後回到場站；限制式(5)確保流量守恆；(6)聯結限制，即同一組收送貨作業要同台車服務；(7)確保服務完此點的車容輛等於上一點服務完的車容量加上要服務此點的需求量；(8)確保優先限制，即車輛服務一顧客需求先收貨再送貨；(9)任一

項作業點完成後立即前往下一個作業點服務，不會出現等待情形；(10)確保車輛服務的過程中不超過車容量限制；(11)決策變數，若車輛從作業點 i 到作業點 j，$x_{ij}^k = 1$；否則$x_{ij}^k = 0$。

PDPTW比PDP多了時間窗限制，如式(12)~(13)，其中式(12)取代了PDP模式中的式(9)，因PDPTW中各個作業點都有時間窗限制，過早到達會導致等待，因此到達下一個作業點的時間應「大於或等於」上一作業點完成時間加上旅行時間，且「小於或等於」上一作業點最晚完成時間。

$$x_{ij}^k\left(T_j^k - T_i^k + s_i + t_{ij}^k\right) \geq 0 \qquad \forall k \in K,\ (i,j) \in A \tag{12}$$

$$e_i \leq T_i^k \leq l_i \qquad \forall k \in K,\ i \in N, \tag{13}$$

2.3 PDPTW 求解方法探討

在PDPTW相關文獻方面，Berbeglia et al.[3]、Sophie et al.[16][17]針對PDP問題進行數學模式、問題分類、求解演算法等有完整回顧。以下將分為啟發式解法、巨集式啟發式解法、最佳解解法與變數產生法做回顧。

2.3.1 啟發式解法 (Heuristic Solution Methods)

在面臨大型問題或具有時效性問題時，採用啟發式解法雖無法確保必能找到全域最佳解，但若設計得宜，通常可求得品質不錯之近似最佳解。以下針對使用啟發式解法來解決PDPTW問題的文獻作一討論說明：

Mitrović-Minić and Laporte[14]使用兩階段啟發式解法解決轉運的PDPTW問題。轉運的PDPTW問題除了包含標準PDPTW的聯結限制與優先限制外，其允許一訂單需求被兩台車服務：一輛車服務收貨點，將收到之貨物運至轉運站，另一輛車至轉運站載貨運至各需求點卸貨。兩階段啟發式解法（two-phase heuristic）包含建構階段和改善階段。於建構階段使用隨機選擇多個起始點的最省插入法，改善階段使用重新插入法。

Lu and Dessouky[12]提出以插入法為基礎的建構啟發式解法解決多車輛的PDPTW問題。其所提出之解法不僅考慮插入時導致的距離增加，尚考慮到因將點插入後導致的時間窗限制。除此之外，作者亦提出一個在插入時的非標準量測方式(Crossing length percentage,CLP)來評估在單一路線中路徑間的交叉程度。CLP定義為所有交叉路徑的總長度除以總路線長度，藉由CLP值評估解的吸引力。最後比較提出之解法與傳統sequential 和parallel插入法，發現提出的解法不論是標準或非標準的量測皆有較佳表現。

表 2 PDPTW 之啟發式解法相關文獻整理

年份	作者	目標式	求解方法
2006	Mitrović-Minić and Laporte[14]	最小化巡迴路徑成本	最省插入法，區域搜尋改善法
2006	Lu and Dessouky[12]	最小化巡迴路徑成本	以插入法為基礎的建構式啟發式解法

2.3.2 巨集式啟發式解法 (Meta-Heuristic)

巨集式啟發式解法能夠跳脫啟發式解法求解過程中陷入局部最佳解的窘境，並改善傳統啟發式解法的求解績效。為了克服傳統啟發式解法在執行上易掉入局部最佳解的缺點，後續研究大部分採用依問題特性設計之合適的巨集式啟發式解法求解問題，針對問題特性設計跳脫局部最佳解的機制，並增加搜尋之廣度與深度。

Nanry and Barnes[15]提出反應式禁忌搜尋法(reactive tabu search)解決PDPTW問題。其目標式為旅行距離、超時與超載之加總，起始解使用貪心插入法(Greedy insertion method)，再使用反應式禁忌搜尋法和三種用來解決優先限制與聯結限制之移步(move)方式，改善起始解。並設計了一套選擇鄰近解的層級(hierarchical search)策略，使其能夠動態地在三種鄰近解中選擇並修正搜尋的方向。作者亦提出一套檢測是否掉入區域最佳解的陷阱，並提供跳脫區域最佳解的方式。最後以Slomon[23]VRPTW之標竿例題為基礎，於目前已知最佳解之相同路徑內將兩點配對，產生適用於PDPTW的標竿例題做驗證測試，並首開先例測試9個100個密集的客戶點，其測試結果不論在求解品質或是求解效率上都有不錯的結果。

Li and Lim[10]提出以禁忌嵌入式模擬退火法(tabu-embedded simulated annealing algorithm)求解多目標的PDPTW問題。目標的優先順序為最小化使用車輛數(M)、總旅行距離(Dist)、總工作時間(ST) (等待時間+服務時間+旅行時間)與司機人員總等待時間(WT)之總和。為了符合目標的優先順序，作者在各項目標前分別加上懲罰值，α、β、γ、λ並訂定$\alpha \gg \beta \gg \gamma \gg \lambda$，使得就算使用建構式所得到之起始解有很多的車輛數，在尋找更好的解時車輛數即會降低。

$$Cost(S) = \alpha M + \beta Dist(S) + \gamma ST(S) + \lambda WT(S) \quad (14)$$

演算法首先使用三種交換法尋找鄰近解：PD-Shift，PD-Exchange與PD-Rearrange，Descent Local Search 找尋區域最佳解。首先用三個定義好的鄰近點為基礎找到區域最佳解，作者將四個目標值建構成禁忌列表以避免產生迴圈，與過去文獻將所存放的edge-move當作禁忌列表不相同。接著以類似多重開始的模擬退火法策略跳脫區域最佳解，並於搜尋時將以求得之且放入禁忌列表以避免產生迴圈。由於過去文獻中，模擬退火法是若無法改善則停止，此篇使用之多重開始的模擬退火法則為若無法改善則從目前為止最好的解再重新開始求解，若重新求解了K次後皆無法再改善才停止。最後以

Slomon[23]VRPTW之標竿例題為基礎，不侷限於在目前已知最佳解之相同路徑內，而是隨機將客戶點配對，使產生之例題更具多樣化分配，產生適用於PDPTW的標竿例題做驗證測試，可求出較少之車輛數。並首開先例測試了56個100個客戶點為多樣化分配的例題，測試結果不論在求解品質或是求解效率上都有不錯的結果。

Lau and Liang [9]提出兩階段法(Two Phase Method)求解最小化使用車輛數與總旅行距離的PDPTW。第一階段結合了傳統插入法與掃瞄法優點之創新的建構啟發式解法產生起始解。第二階段使用禁忌搜尋法改善目標值，並透過三種不同鄰近區域移動方式尋找最佳解。此篇亦提出一將Slomon[23]的VRPTW之標竿例題轉為有好的解的PDPTW標竿例題。由於PDPTW比VRPTW多了優先順序與配對限制，在轉換時必須注意是否仍是可行解與轉換後仍保持好的解。

最後Lau and Liang [9]提出一轉換方法產生包含四種類型的27個例題作測試。測試分為兩部分，第一部分比較插入法、分割插入法、掃描法與目前已知最好的解，測試結果發現分割插入法在27個例題中有18個例題為最好的解，插入法和分割插入法都可得到最好的車輛數，分割插入法尚可得到較好的旅行距離。第二部份比較作者提出之禁忌搜尋法和目前已知最好的解，測試結果發現使用作者提出之禁忌搜尋法對大多數的例題可產生與目前已知最好的解之相近解。

Bent and Hentenreyck[2]點出降低使用車輛數是降低總旅行成本的關鍵之一，使用兩階段啟發式演算法求解PDPTW，第一階段使用簡易的模擬退火法(simulated annealing, SA)降低車輛數，第二階段使用大型鄰近搜尋法(Large Neighborhood Search, LNS)降低總旅行成本。最後使用Li and Lim[10]產生之PDPTW之標竿例題測試其績效，於100個客戶點突破兩題、200個客戶點突破28題，600個客戶點亦突破46題的解。

本研究將有關PDPTW問題之巨集式啟發式解法的相關文獻，依其問題型態、求解方法與測試範利來源分類整理如表3所示。雖然巨集式啟發式解法已可在有效時間內求得不錯之解，但巨集式啟發式解法相對於最佳解仍然有一段距離。

表 3 PDPTW 之巨集式啟發式解法相關文獻整理

年份	作者	目標式	求解方法	測試範例
2000	Nanry and Barnes [15]	最小化巡迴路徑成本	反應式禁忌搜尋法，貪心插入法	以 Slomon[23]VRPTW之標竿例題為基礎，於目前已知最佳解之相同路徑內將兩點配對
2001	Li and Lim [10]	最小化巡迴路徑成本、車輛數、等待時間、工作時間	禁忌嵌入式模擬退火法	以 Slomon[23]VRPTW之標竿例題為基礎，隨機將客戶點配對
2002	Lau and Liang [9]	最小化巡迴路徑成本、車輛數	插入法與掃瞄法禁忌搜尋法	以 Slomon[23]VRPTW之標竿例題為基礎，提

				出一轉換方式，其轉換時注意是否保持可行解
2006	Bent and Hentenryck[2]	最小化巡迴路徑成本、車輛數	模擬退火法 大型鄰近搜尋法	使用 Li and Lim[10]產生之 PDPTW 之標竿例題

2.3.3 變數產生法

變數產生法由Dantzig and Wolfe[4]所提出，又稱Dantzig-Wolfe Decomposition，主要概念是利用線性規劃中的對偶理論(Dual Theory)產生可改善目標值的變數(column)，以避免浪費時間於窮舉對問題求解沒有貢獻的變數。因此，變數產生法常用於解決具有龐大變數的線性規劃問題，例如集合分割問題(Set Partitioning Problem)與集合涵蓋問題(Set Covering Problem)。假設有一集合分割問題如下：

$$
\begin{aligned}
&Minimize \sum_{r\in R} c_r x_r \\
&\text{Subject to } \sum_{r\in R} a_{ir} x_r = 1 \qquad \forall i \in N \\
&x_r \in \{0,1\} \qquad r \in R
\end{aligned}
\tag{15}
$$

變數產生法將上述集合分割問題分為主問題(Master Problem)與子問題(Sub Problem)。由於考量求解的可行性與方便性，求解集合涵蓋問題較求解集合分割問題容易，故多數研究將欲求解之集合分割問題放鬆，改求解集合涵蓋問題，並放鬆整數限制為線性規畫問題，之後再做適當的調整機制以滿足集合分割之整數限制。因此放鬆(15)為集合涵蓋問題並放鬆整數限制式，使變數產生法中的主問題成為放鬆整數限制的集合涵蓋問題，即為一線性規劃問題(16)。利用簡捷法(Simplex Method)求解主問題目前所包含變數的最佳解。並依據對偶理論(Dual Theory)，定義線性規劃問題(16)限制式之對偶變數為π_i，主問題之對偶問題(Dual Problem)如(17)所示：

(Primal)

$$
\begin{aligned}
&Min. \sum_{r\in R} c_r x_r \\
&\text{S.T} \quad \sum_{r\in R} a_{ir} x_r \geq 1 \qquad \forall i \in N \\
&\qquad x_r \geq 0 \qquad \forall r \in R
\end{aligned}
\tag{16}
$$

(Dual)

$$
\begin{aligned}
&Max. \sum_{i\in N} \pi_i \\
&\qquad \sum_{r\in N} a_{ir} \pi_i \leq c_r \qquad \forall r \in R \\
&\qquad \pi_i \geq 0 \qquad \forall i \in N
\end{aligned}
\tag{17}
$$

建構子問題產生能對主問題目標值有改善的變數(columns)。子問題通常構建為最短路徑問題(Shortest Path Problem)或有限制的最短路徑問題(constrained shortest path problem)。將主問題得到的對偶變數值傳入子問題中，求解子問題，若求解得知的變數可改善目標值，則加入主問題中重新求解。重複以上的步驟，逐步改善主問題的解，直到無法再產生可改善目標值的變數為止，即得到最佳解。判斷子問題求得的變數是否可以改善目標值，主要是依據對偶可行性(Dual feasibility)的觀念。

根據對偶可行性可知：若可證明對尚未考慮的變數(columns)均滿足(18)則可知剩餘尚未考慮之變數無法再改善目標值的品質，即找到最佳解；否則可從尚未考慮的變數中找到一個新的變數，使路徑r的減少成本$C_r^{'} = C_r - \sum_{i\epsilon N} a_{ir}\pi_i < 0$，加入主問題中重新求解。

$$C_r^{'} = C_r - \sum_{i\epsilon N} a_{ir}\pi_i \geq 0 \ \forall r \in R \tag{18}$$

若最後求得之最佳解為整數解，此解即為原集合涵蓋問題之最佳解；若不為整數解，多數文獻採用分支定限法以求得整數解。

變數產生法之優點在於求解過程中每次僅考慮部分變數，透過子問題產生一些對主問題可改善目標值的變數，以增進求解效率。過去研究已將變數產生法應用於PDPTW問題。

Dumas et al.[7]利用變數產生法求解多台車的PDPTW問題，並考慮異種車隊與多場站等限制，其將子問題建構成一有限制式最短路徑問題，並使用forward動態規劃演算法求解子問題，並提出label domination 和label elimination 規則，此篇提出之演算法可解決至少50個需求任務。

Xu et al.[26]使用以變數產生法為基礎的啟發式演算法解決多車輛的PDPTW問題。此研究考量了每個起迄點有多時間窗限制、載重限制、司機工時限制等額外限制式。變數產生法的主問題建構為IP數學模式，子問題使用兩個啟發式解法：合併(merge) 和兩階段(two-phase)啟發式解法，兩階段包含合併路線和需求點的插入與刪除。此篇提出之演算法可求解隨機產生的200個需求任務。

Sigurd et al.[22]探討運輸豬的問題，每個需求任務代表運輸動物從一個地點運送到另一特定地點，為了避免擴散(健康的豬必定不能被曾經載運過有病豬的車所運送)此問題有額外的優先限制。Sigurd 將子問題設計成一非環狀的層級圖，可快速的解決較大的例子。此篇提出之演算法可求解580個需求任務。

Ropke and Cordeau[18]用Branch and Price演算法加上有效不等式求解PDPTW問題。其子問題構建為有限制式的最短路徑問題，使用label setting 最短路徑演算法求解。有效不等式加入後使子問題產生的影響亦於此篇中討論。最後研究以Slomon所提出之產生PDPTW標竿例題方法，產生適合此篇問題特性之PDPTW例題做測試。

表 4 PDPTW 之最佳解解法相關文獻整理

年份	作者	目標式	求解方法
1991	Dumas et al.[7]	最小化巡迴路徑成本	變數產生法 有限制式的最短路徑演算法
2003	Xu et al.[26]	最小化巡迴路徑成本	變數產生法 啟發式解法 動態規劃
2004	Sigurd et al[22]	最小化巡迴路徑成本	變數產生法 非環狀的層級圖
2007	Ropke and Cordeau[18]	最小化巡迴路徑成本	分支裁切價格法

2.4 小結

綜合上述，多數研究使用具有跳脫區域最佳解機制之啟發式解法，可求得品質不錯的解，並可解決規模較大的問題，或在有限時間內求得一組可行解。但跳脫區域最佳解之機制無法保證接近最佳解，故若僅為求解速度而使用啟發式解法，不僅無法證明求得之解是否接近最佳解，且無法保證跳脫區域最佳解機制可應用於任何情況。故對於不需快速的求解時間，則可考慮使用以最佳解為基礎的演算法求解。

本研究為替貨運業之預先派遣作業規劃－收送貨路徑規劃方法，將不採用啟發式解法，而使用最佳解為基礎之演算法求解PDPTW，並設計變數產生法中子問題求解演算法，使其可有效找出對主問題有貢獻之變數，快速解決較大規模之網路。

第三章　數學模式與求解演算法

本研究首先對有時間窗限制的收送貨問題建構數學模式，再使用以最佳解為基礎的變數產生法(Column Generation)求解數學模式，期望能針對預先調度派遣問題提出車輛的巡迴路徑組合，以能供調度者參考使用，提升貨運公司之整體營運效率。本章將針對所探討的問題建構數學模式與求解方法。3.1節對有時間窗限制的收送貨問題建構數學模式；3.2節將說明本研究之演算法求解流程；3.3節說明演算法中之變數產生法；3.4節說明處理重複涵蓋之啟發式解法。

3.1 有時間窗限制的收送貨問題模式

本研究將有時間窗限制的收送貨問題建構成集合分割問題(Set Partitioning Problem)，其問題定式如下：

$$Min \sum_{r \in R} c_r x_r \tag{19}$$

$$\text{ST} \quad \sum_{r \in R} a_{ir} x_r = 1 \qquad \forall i \in N \tag{20}$$

$$\sum_{r \in R} x_r \le K \tag{21}$$

$$x_r \in \{0,1\} \qquad r \in R \tag{22}$$

其中變數x_r為二元變數，代表一可行之路徑(column)，當路徑 r 被選到時x_r=1，反之x_r=0；R 為所有路徑之集合。C_r 為路徑 r 的旅行距離成本；N 為所有任務點的集合，包含收貨作業與送貨作業任務。目標式(19)為最小化所有路徑的總旅行距離成本；限制式(20)為集合分割限制式，表每一任務 i 都需被服恰一次。若路徑 r 包含任務 i，則 a_{ir}=1，反之 a_{ir}=0。限制式(21)為車輛數限制，K 為可使用車輛數上限。

可行路徑定義為在一路徑中需滿足各任務之時間窗限制、優先限制、聯結限制與行車過程中的車容量上限。

3.2 演算法求解流程

本研究將PDPTW問題建構為集合分割問題，將其放鬆整數限制成為一線性規劃問題，由於集合涵蓋問題較集合分割問題好求解，故將變數產生法中的主問題(Master Problem)建構為放鬆整數限制後的集合涵蓋問題。之後產生一組主問題的起始可行解，即為受限主問題(Restricted Master Problem)，利用簡捷法(Simplex Method)求解受限主問題以獲得對偶變數值。將子問題設計成有限制式的最短路徑問題，將從受限主問題中求解得知之對偶變數值代入路網之節線成本，並發展一個以Dijkstra's演算法為基礎的最短路徑演算法。求解出的最短路徑即代表主問題中的一個可行變數，藉由對偶可行性判斷

是否將此變數加入主問題中繼續求解，若不滿足對偶可行性則加入主問題，反之；則表示已獲得主問題放鬆後的最佳解,停止變數產生法的流程。由於主問題放鬆了整數限制，故藉由變數產生法所產生的解可能為非整數解，此時再利用分支定限法求得整數解，即為集合涵蓋之解，故有重複涵蓋之問題，本研究再使用移除法解決重覆涵蓋問題。圖4為演算法流程圖。此求解演算法稱為演算法DECAP(Dijkstra's Embedded Column Generation Algorithm for Solving PDPTW)。

1. 資料輸入

依據問題特性描述，設定所有收送貨任務、貨車容量上限、各任務可服務時間範圍等資訊。

2. 將 PDPTW 建構為集合分割問題，如 3.1 節所示。

3. 將變數產生法之主問題建構為線性規劃問題。

即將集合分割問題放鬆為集合涵蓋與整數限制，主問題之定式將於 3.3.1 介紹。

4. 產生一組可行的起始變數集合代入主問題中，成為受限主問題。

設定每一筆訂單就是一條路徑，即每輛車僅經過一對收貨點與送貨點。

5. 利用簡捷法求得目前受限主問題的最佳解與對應之對偶變數值

利用求解線性規劃之軟體求解目前主問題之最佳解，並獲得對應之對偶變數值，即對應任務 i 之對偶值。

6. 建構子問題為最短路徑問題

本研究針對子問題建構成一有限制式的最短路徑問題，求解出的最短路徑即代表主問題中的一個變數，之後依據對偶可行性判斷是否可將此新變數加入主問題中繼續求解，以獲得更好的解。

7. 定義子問題之路網

定義子問題路網的節點、節線與節線成本，並產生路網。

8. 將對偶變數值傳入子問題路網中

將由求解線性規劃之軟體求得的對偶變數值π_i傳入子問題路網中，π_i為對應之對偶變數，即路網中任務 i 的對偶變數值。

9. 求解子問題

本研究所設計的子路網問題需滿足時間窗限制、車輛容量限制、優先限制、聯結限制等，因此欲找尋最短路徑時，並非傳統的最短路徑演算法即可解決。因此本研究提出兩種求解子問題之演算法，使之能在滿足各限制下找出總節線成本最小的路徑。

10. 判斷是否滿足對偶可行性

利用對偶理論中的對偶可行性判斷子問題求得之最短路徑對於主問題是否有貢獻度,若滿足對偶可行性則得知已達最佳解;反之則代表此路徑對主問題目標式有貢獻，將對應此路徑的變數加入主問題，至步驟 5 繼續求解。

11. 判斷目前受限主問題的最佳解是否為整數解

若目前受限主問題的最佳解即為整數解，則已找到主問題之最佳解；反之，則利用分支定限法(Branch and Bound)求得整數解。

12. 判斷求解結果是否有重覆涵蓋的不可行解

若有重覆涵蓋之不可行解，則利用啟發式解法之移除法解決不可行解之情形，以獲得車輛路徑巡迴組合；若無，則求解結果即為 PDPTW 所求之車輛路徑巡迴組合。

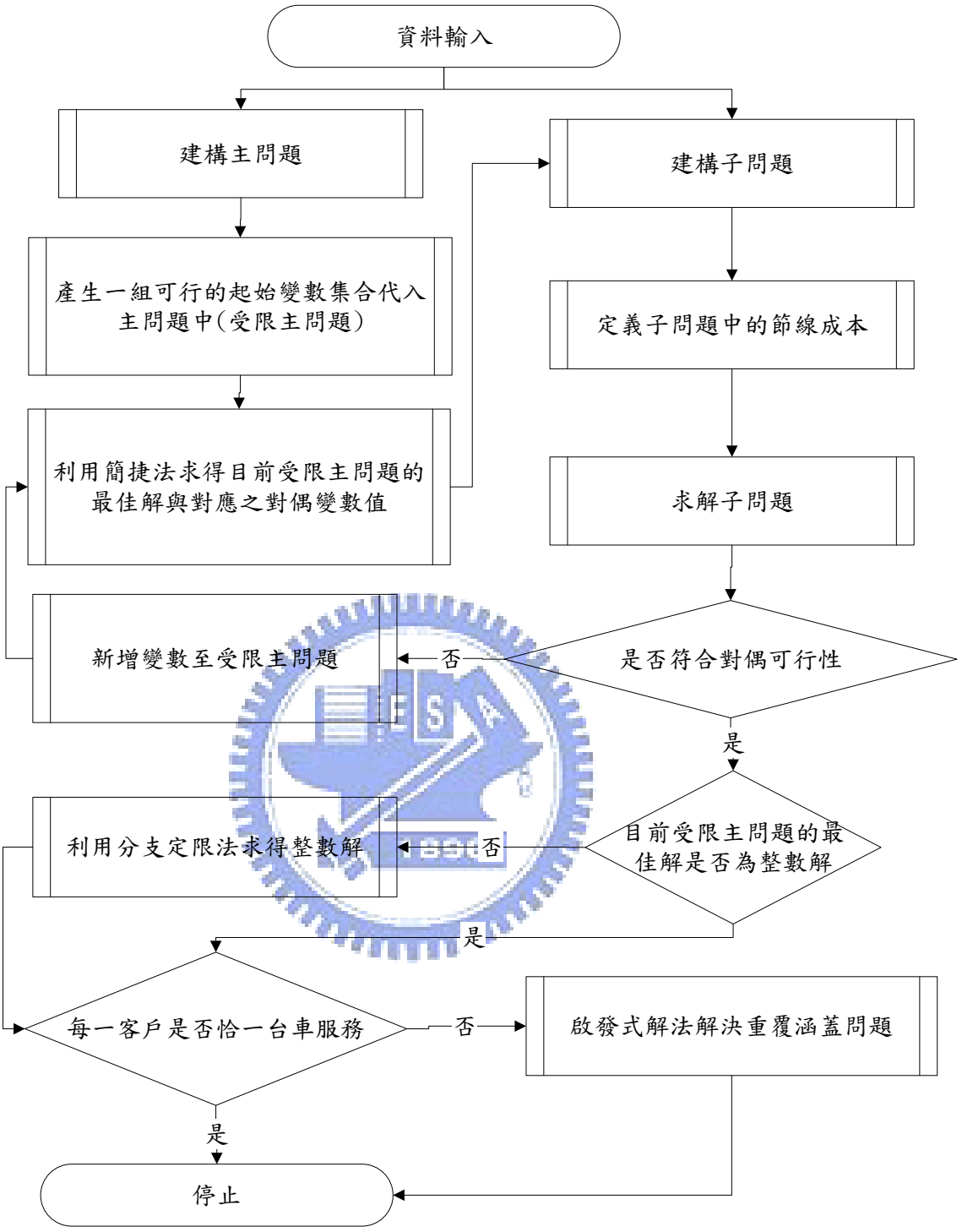

圖 4 演算法 DECAP 求解流程圖

3.3 變數產生法(Column Generation)

3.3.1 主問題(Master Problem)

由3.1節可知，本研究將有時間窗限制的收送貨問題建構為集合分割問題，使所有任務點皆能被服務恰好一次。應用變數產生法時，主問題定式僅需放鬆本研究建構之集合分割問題之整數限制，使其成為線性規劃問題即可。但因集合涵蓋問題較集合分割問題易求解，且除非一任務由兩台車服務之成本較一台車服務之成本小，否則重複涵蓋情形不會發生，故本研究將變數生法之主問題建構為放鬆整數限制後的集合涵蓋問題，即為一線性規劃問題。主問題定式如下：

$$Min.\sum_{r\in R} c_r x_r \tag{23}$$

$$\text{S.T} \quad \sum_{r\in R} a_{ir} x_r \geq 1 \qquad \forall i \in N \tag{24}$$

$$\sum_{r\in R} x_r \leq K \tag{25}$$

$$x_r \geq 0 \qquad \forall r \in R \tag{26}$$

其中限制式(24)即為集合涵蓋問題之限制式，表每一任務i皆至少被服務一次。(26)式即為放鬆整數限制後之非負限制。

3.3.2 受限主問題(Restricted Master Problem)

由於主問題所包含之路徑組合(R)數量龐大，故在求解主問題前先產生一組包含起始可行解之變數，僅包含起始可行解變數之主問題即稱為受限主問題(Restricted Master Problem)。本研究以同一訂單需求為一條路徑視為主問題之起始可行變數，即一條路徑包含一收送貨配對。為使起始之受限主問題為可行解，本研究先設定K為訂單數目，當受限主問題之變數增加到一定數量時，將K設定回該公司之車輛數，以滿足該公司車輛數限制。受限主問題之定式如下：

$$Min\sum_{r\in J} c_r x_r \tag{27}$$

$$\text{S.T} \quad \sum_{r\in J} a_{ir} x_r \geq 1 \qquad \forall i \in N \tag{28}$$

$$\sum_{r\in J} x_r \leq K \tag{29}$$

$$x_r \geq 0 \qquad \forall r \in J \tag{30}$$

其中J為已使用變數之集合，$J<<R$。

3.3.3 子問題(SubProblem)

由於將主問題建構為一線性規劃問題，故可利用簡捷法(Simplex Method)求解受限主問題的最佳解，與其所對應的對偶變數值。並依據對偶理論(Dual Theory)，定義受限主問題之限制式(28)對應之對偶變數為π_i，限制式(29)對應之對偶變數為π_0，受主問題之對偶問題(Dual Problem)如(31)所示：

$$Max \sum_{i \in N} \pi_i + K\pi_0$$

$$\text{S.T} \quad \sum_{i \in N} a_{ir}\pi_i + \pi_0 \le c_r \qquad \forall r \in J$$

$$\pi_i \ge 0 \qquad \forall i \in N$$

$$\pi_0 \le 0 \tag{31}$$

根據對偶可行性(Dual Feasibility)可知：若可證明對尚未考慮的變數(columns)均滿足(32)則可知剩餘尚未考慮之變數無法再改善目標值的品質，即找到最佳解；否則可從尚未考慮的變數中找到一個可使路徑r的減少成本(Reduced Cost) $C_r^{'} = C_r - \sum_{i \in N} a_{ir}\pi_i - \pi_0 < 0$的新變數，加入主問題中重新求解。

$$C_r^{'} = C_r - \sum_{i \in N} a_{ir}\pi_i - \pi_0 \ge 0 \quad \forall r \in R \tag{32}$$

欲找一減少成本小於零之路徑 r，可視為求解滿足 PDPTW 特性之目標函式 $Min\ C_r^{'}$。故子問題建構為滿足 PDPTW 限制之最短路徑問題。若最後求得之最佳解為整數解，此解即為原集合涵蓋問題之最佳解；若不為整數解，多數文獻採用分支定限法(Branch and Bound)以求得整數解。由於一條路徑之旅行距離成本等於該路徑所有經過之節線加總，故C_r=$\sum_{(i,j) \epsilon A} d_{ij} x_{ij}^r$，其中 A 為所有節線的集合，d_{ij} 為節點 i 到節點到節點 j 的旅行距離，x_{ij}^r 為決策變數，若路徑 r 經節線(i,j)，$x_{ij}^r = 1$；否則$x_{ij}^r = 0$。故目標函式可改寫成(33)。

$$Min\ C_r^{'} = \sum_{(i,j) \epsilon A} d_{ij} x_{ij}^r - \sum_{i \in N} a_{ir}\pi_i - \pi_0 \tag{33}$$

3.3.4 建構子問題路網

根據變數產生法的概念，透過子問題的構建產生適合的變數加入主問題，以改善解的品質。因此，本節構建收送貨之時空路網，並以一範例為例，以構建一最短路徑，期望找到能改善主問題目標值之路徑(columns)。

(一) 路網結構

假設圖 G 為一路網，V 表示節點的集合，A 為節線的集合。因此，G 可以定義為 $G = (V, A)$。茲將其路網結構詳述如下：

1. 節點(node)

(1) 令 n 為訂單數，則節點集合 V 包含起點(0)、各訂單的收貨任務點N^+={1,2,…,n}與各訂單的送貨任務點N^-={n+1,n+2,...,2n}與迄點(2n+1)，訂單需求集合 $N = \{N^+ \cup N^-\}$。

(2) 每一訂單必有一收貨任務與一送貨任務，故每個收貨任務必有一相對應之送貨任務與之配對(pairing constraint)。第 r 筆訂單之收貨任務即為節點集合 N 中之 r，送貨任務即為 n+r。

(3) 每個節點有數個屬性：節點編號 i、最早可開始執行之時間 e_i、最晚需開始執行之時間 l_i、需求載重量 q_i、對偶變數值π_i。若節點為收貨任務，則需求載重量為正值；反之，若節點為送貨任務，則需求載重量為負值。

2. 節線(Arcs)

子問題路網中，每一任務皆以一節點表示，節線為兩不同節點相連所組成，故$(i,j) \in A$，其中$i,j \in V$且$i \neq j$。節點分類包含收貨任務點、送貨任務點、起點與迄點，各類節點間相互連接形成節線之條件如下所述：

(1) 起點(0)與所有收貨點相連接，迄點(2n+1)與所有送貨點相連。

(2) 任兩任務節點 i 與 j 間需符合$e_i + t_{ij} \leq l_j$條件即可相連。其中 t_{ij} 為任務節點 i 到任務節點 j 的旅行時間。

(3) 為保持優先限制，同一配對任務不能先送貨再收貨 (n+i, i), (2n+1,0),(2n+1,i)(2n+1,n+i),i=1,...,n 等節線不可相連。

3. 節線成本(Cost)

由於變數產生法是依據簡捷法之對偶可行性(Dual Feasibility)判斷目前主問題是否已找到最佳解，因此子問題即為在路網中產生最短路徑；路徑成本為經過之節線成本加總，因此路網中節線成本加總須滿足對偶可行性條件，即(32)所示。但子問題路網中，對偶變數對應於節點上，因此定義節線成本時，需將對偶變數轉移至節線上。

令節點 i 到節點 j 的節線成本定義為$C'_{ij} = d_{ij} - \pi_j$，其中d_{ij}為節點 i 到節點 j 的旅行距離，π_j 為節點 j 所對應之對偶變數值。π_0於路網中則視為迄點之對偶變數值。

因此，根據本研究所定義之節線成本，將子問題路網之節線成本修正為旅行距離成本減去對偶成本之形式，路徑 r 的成本即為所經過之節線成本加總，亦即 $\sum_{(i,j)\in A} d_{ij}\, x_{ij}^{r} - \sum_{j\in V} \pi_j\, a_{jr}$。因此路徑 r 的成本可寫成式(33)，其中 x_{ij}^{r} 為決策變數，若節點 i 到節點 j 屬於路徑 r 為 1；否則為 0。由式(33)可知路徑 r 的成本與主問題中路徑 r 的減少成本相等，故可知由子問題中求得之最短路徑成本，即為該路徑於對偶問題中對偶可行性限制式的左邊項目。

$$
\begin{aligned}
C_r' &= \sum_{(i,j)\epsilon A} C_{ij}' x_{ij}^{r} \\
&= \sum_{(i,j)\in A} d_{ij}\, x_{ij}^{r} - \sum_{j\in V} \pi_j\, a_{jr} \\
&= \sum_{(i,j)\in A} x_{ij}^{r}\, d_{ij} - \sum_{i\epsilon N} a_{ir}\pi_i - \pi_0
\end{aligned}
\tag{33}
$$

(二) 建構子問題網路範例

以一個含有 8 個節點(3 筆訂單)的收送貨路網為範例，說明子問題路網之構建過程。假設車容量上限為 30，節點資訊如表 5 所示，節點編號 i、最早可執行時間 e_i、最晚可執行時間 l_i 與需求載重量 q_i 皆為固定值；對偶變數值因由主問題求解所得，故每次加入新的可行路徑後(column)，求解主問題所得之對偶變數值會不同。

根據本節節點之定義，表 5 中之節點編號 0 與節點編號 7 分別為起迄點；節點編號 1~3 為收貨任務點，需求載重量為正值；節點編號 4~6 為送貨任務點，需求載重量為負值。節點編號 1、4 為同一訂單；節點編號 2、5 為同一訂單；同理，節點編號 3、6 為同一訂單。圖 5 圖 6 之節點內數字為節點編號，以黑色節點表示起迄點；白色節點表示收貨點；灰色節點表示送貨點。本範例設定起始可行變數(任務)為同一訂單為一條路徑，故起始解如圖 5 所示，起始解為 0→1→4→7、0→2→5→7 和 0→3→6→7。表 5 中之對偶變數值即由求解起始的主問題後可得。

表 5 屬性資訊表

節點編號 i	最早可執行時間 e_i	最晚可執行時間 l_i	需求載重量 q_i	對偶變數值 π_i
0	0	230	0	0
1	37	47	16	0
2	97	107	16	0
3	127	137	9	0
4	71	81	-16	50
5	0	193	-16	64
6	0	188	-9	79
7	0	10000	0	0

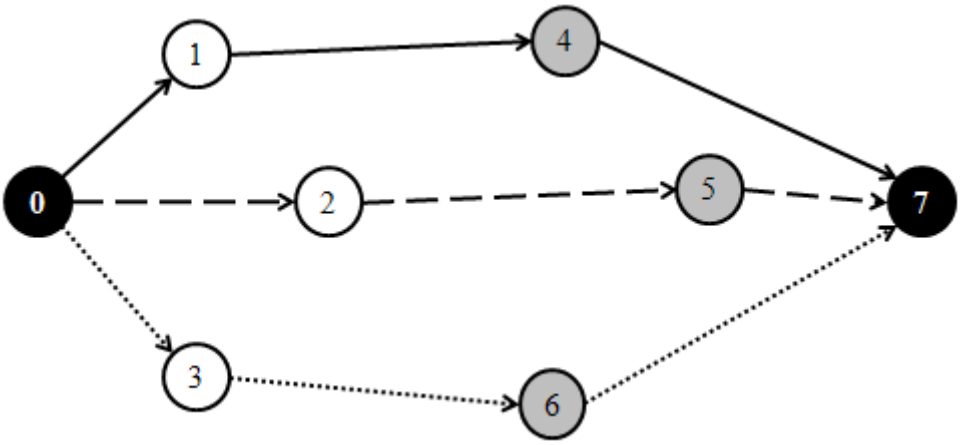

圖 5 起始可行變數示意圖

透過節點與節線之定義，可設定節點之間的連接情況，及各節線之對應成本。如圖 6 所示，中括號為時間窗$[e_i, l_i]$，小括號為需求載重量(q_i)，以斜體字加底線書寫則為本範例設定之起始解所求得之對偶變數值$(\pi_i \text{與} \pi_0)$。假設旅行速度保持為 1m/s，則各節點間之旅行時間等於旅行距離，已知各節點間之旅行時間(t_{ij})等於旅行距離(d_{ij})，即為圖 6 節線上之數字，則可藉由時間窗限制$(e_i + t_{ij} \leq l_j)$、優先限制，與聯結限制構建出一有向路網。而各節線所對應之成本則可藉由$C'_{ij} = d_{ij} - \pi_j$求得。

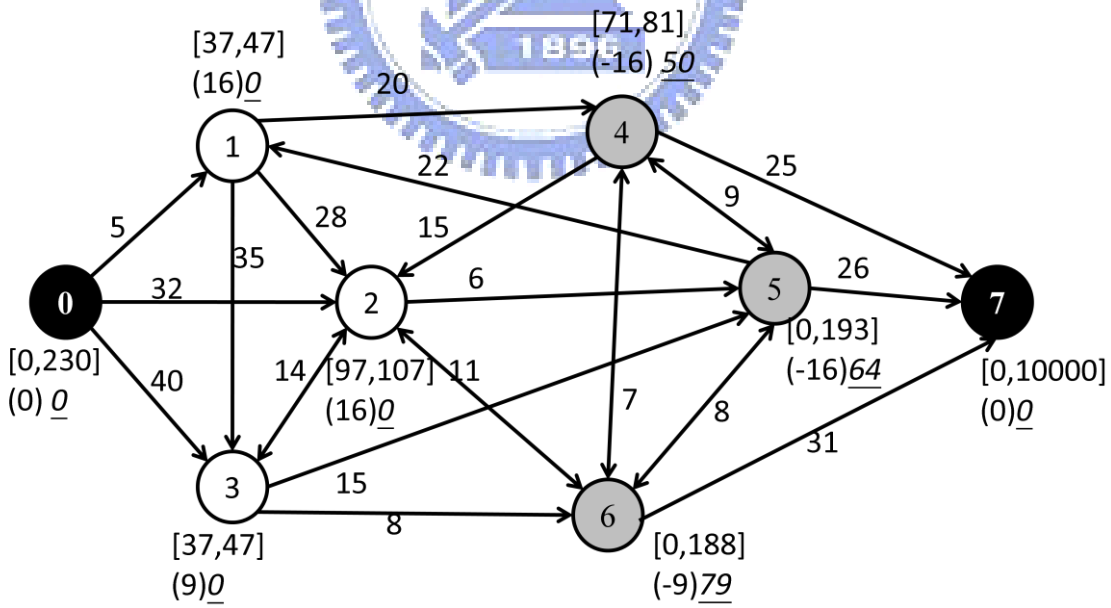

圖 6 子問題路網建構圖

3.3.5 求解子問題演算法

由 3.3.3 節可知，子問題為有限制之最短路徑問題，本研究應用 Dijkstra's Algorithm 求解最短路徑問題。Dijkstra's Algorithm 於每次迭代(iteration)時，尋找標籤尚未被固定且成本最小的節點，以此點更新其標籤尚未被固定的下游節點。若下游點之成本標籤值比原成本還小時，則將其成本標籤與其前置點更新，並將該節點標籤固定。

Dijkstra's Algorithm 不能應用於有負成本節線之網路上，因 Dijkstra's Algorithm 之觀念為每次選成本最小之節線，若節線成本有負，則所得到之最短路徑並非真實之最短路徑。假設欲找一條從節點 S 到節點 D 之最短路徑，如圖 7 所示，迭代 1 時，將固定節點 S，更新節點 1 的成本為 3 與節點 D 的成本為 2；迭代 2 時，固定成本最小的節點 D，D 無其他可更新點；迭代 3，固定標籤尚為被固定且成本最小的節點 1，從節點 1 至節點 D，因成本為負值，故到達節點 D 之成本為 1，比節點 D 已被固定之成本值還小。故 Dijkstra's Algorithm 於負成本節線之網路不能找到最短路徑。

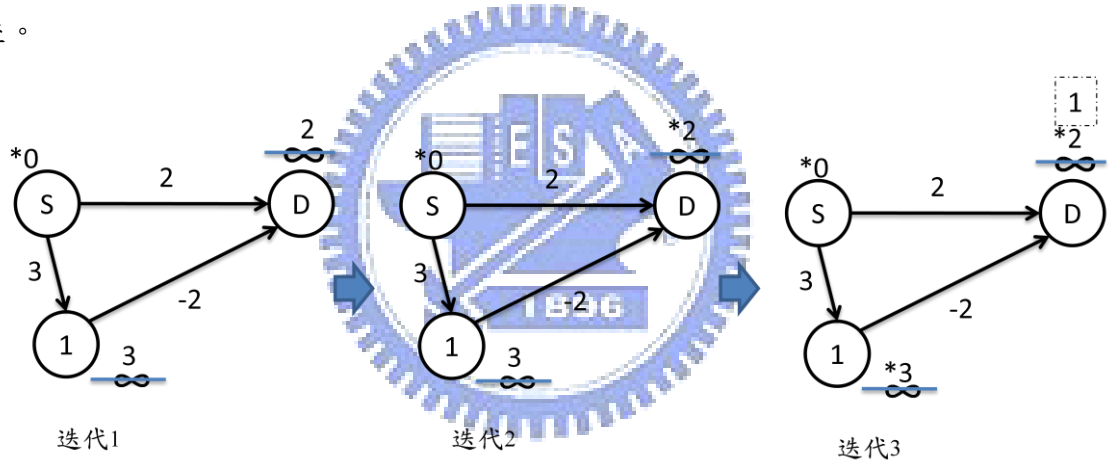

圖 7 成本為負之 Dijkstra's Algorithm

本研究之節線成本定義為節線距離減該點之對偶變數值，故節線可能有負成本之情形，但本研究之子問題非單純之最短路徑，為有限制之最短路徑問題，除了需考量成本外，尚有其他限制須考量，如時間窗限制、優先限制、聯結限制與車容量限制等，Dijkstra's Algorithm 無法直接適用於本研究。有負成本節線之網路於有限制條件下仍有機會求得真實之最短路徑，例如於圖 7 之迭代 3，從節點 1 至節點 D 雖比已被固定之成本標籤小，但卻因為時間窗限制，無法到達節點 D，故從 S 到 D 點仍是最短路徑。因此本研究應用多重標籤觀念[7]，修正 Dijkstra's Algorithm 求解子問題。

本研究提出修正後之 Dijkstra's 演算法為考量多種收送貨因素下的最短路徑問題，故除了考量傳統 Dijkstra's 演算法之「成本 *Ci*」與「前置點 *Pi*」兩標籤外，尚包含用來記錄收送貨問題限制情形的標籤，其包含有：「執行時間 *Ti*」、「累積載重 *Li*」，與「已收未送的訂單集合 $PickupSet_i$」等三個標籤，成為多重標籤 Dijkstra's 演算法。其中「執

行時間 *Ti*」記錄車輛開始執行節點 *i* 之時間；「累積載重 *Li*」記錄執行完節點 *i* 後車輛累積載重量；「已收未送的訂單集合 $PickupSet_i$」記錄到執行完節點 *i* 後，已收取貨物但尚未將其貨物送達送貨點之訂單(聯結限制)。

(一) 演算法 DECAP 之求解子問題演算法流程

本研究修正傳統Dijkstra's更新節點之檢驗條件為演算法A，由於演算法A包含了許多限制條件，故有迄點無上游點之缺點，因此本研究再提出改善之演算法B，演算法B包含演算法B1與演算法B2，將演算法A與演算法B結合，即為本研究演算法DECAP求解子問題之總演算法流程。如圖8所示。詳細步驟詳述如後。

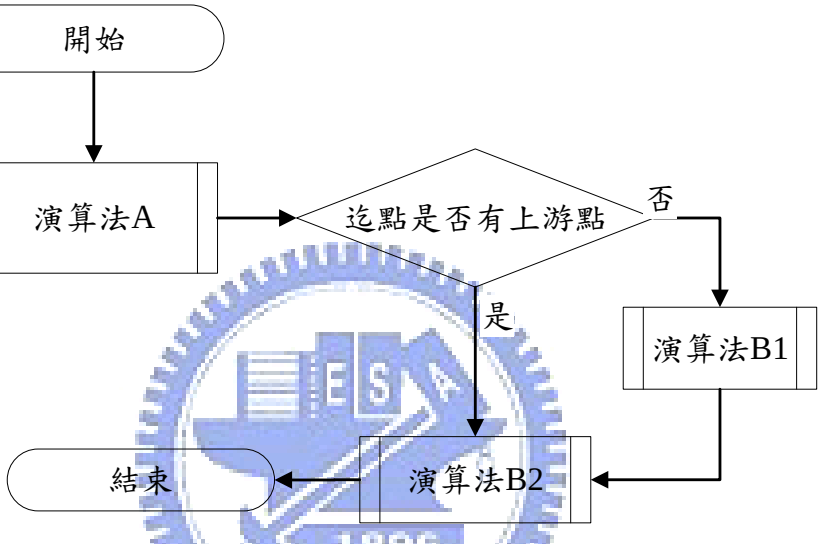

圖 8 求解子問題之總演算法流程圖

(二) 演算法 A

本研究修正傳統 Dijkstra's 演算法更新節點之檢驗條件為演算法 A，修正方式與演算步驟如下。

1. 修正傳統 Dijkstra's 演算法更新節點之檢驗條件

演算法 A 在判斷是否更新每個節點之多重標籤前，除了與傳統 Dijkstra's 演算法相同需檢查成本外，亦需針對各節點做下列五種限制之判斷，使之能找到一條滿足收送貨情形之最短路徑。若其一限制違反則停止檢查，不更新節點上之多重標籤，並繼續檢查下一點。此五種限制如下：

(1). 優先限制：針對同一筆訂單，車輛需先執行訂單之收貨任務才可執行該訂單之送貨任務。

(2). 聯結限制：每一訂單之收送貨任務均由同一輛車服務。

(3). 時間窗限制：若車輛在某節點 *i* 之最早可執行時間(e_i)前到達，則車輛必須等

候至最早時間(e_i)才可開始執行任務；若車輛在某節點之最早可執行時間(e_i)與最晚可執行時間(l_i)之間到達，則車輛可直接執行任務；若車輛在某節點之最晚可執行時間(l_i)後到達，則該車輛不可執行該節點。

(4). 車輛容量限制：若車輛於某一節點之累積載重超過車輛容量(Cap)，則該車輛不可執行該節點。

(5). 收送貨合理性：車輛之累積載重若不等於零，則車輛不允許回迄點。

2. 演算法 A 之執行步驟

演算法 A 步驟如下，流程圖如圖 9 所示。令 S 為標籤被固定住的節點集合。為分辨每一節點於每一迭代是否已試著更新，故將每一點分類為已嘗試或未嘗試。

步驟A-1. (多重標籤初始化)設定所有點的「前置點」為零、「累積載重」為零、起點的「執行時間」為零，其他點的「執行時間」為無窮大，起點的「累積成本」為零，其他點的「累積成本」為無窮大，所有點的「已收未送的訂單集合」設為空集合。標籤被固定的節點集合設為空集合(S={∅})；所有點設為未嘗試。

步驟A-2. (停止條件)若迄點已在 S 中，則演算法停止。反之，到步驟 A-3。

步驟A-3. 從標籤尚未被固定的點中，找成本最小的點，設為 u。並將 u 的標籤固定，u 放入 S 中。

步驟A-4. 對所有與 u 相連的節點 $j \epsilon N$，計算節線成本 $C_{uj} = d_{uj} - \pi_j$。

步驟A-5. 判斷節點 u 是否有與其相鄰(u 之下游點)，且未嘗試經過的點。若有，則設該點為 v，到步驟 A-6；若無，則將所有點設為未嘗試，到步驟 A-2。

步驟A-6. (優先限制)若節點 v 為收貨點，則滿足優先限制，到步驟 A-7。若節點 v 為送貨點，則判斷節點 v 的收貨點是否在節點 u 的「已收未送的訂單集合」中，若在，則滿足優先限制，到步驟 A-7；若不在，則違反優先限制，將節點 v 設為已嘗試，到步驟 A-5。

步驟A-7. (時間窗限制)若從節點 u 到達節點 v 的時間等於或早於節點 v 最晚可執行時間($T_u + t_{uv} \leq l_v$)，則滿足時間窗限制，到步驟 A-8；反之，若從節點 u 到達節點 v 的時間大於節點 v 最晚可執行時間($T_u + t_{uv} > l_v$)，則違反時間窗限制，將節點 v 設為已嘗試，回到步驟 A-5。

步驟A-8. (車輛容量限制)若節點 v 為送貨點，則車上總載重減少，必滿足車輛容量限制，到步驟 A-9；若節點 v 為收貨點，車上總載重將增加，故若節點 u 的累積載重與節點 v 的需求載重量加總大於車輛容量($L_u + q_v > C\text{ap}$)，則違反車輛容量限制，將節點 v 設為已嘗試，回到步驟 A-5；反之，若加總小於等於

車容量($L_u + q_v \leq Cap$)，則滿足車容量限制，到步驟 A-9。

步驟A-9. (成本限制)若節點 v 之成本比該點原本成本小($C_u + C_{uv} < C_v$)，則到步驟 A-10；否則，將節點 v 設為已嘗試，回到步驟 A-5。

步驟A-10. (將節點 v 放入 S 中，並更新節點 v 的多重標籤)更新節點 v 的「前置點」為 P_v=u；節點 v 的「到達時間」為 T_v =$Max(T_u + t_{uv}, l_v)$；到節點 v 車輛的「累積載重量」為$L_v = L_u + q_v$；到 v 點「累積成本」為$C_v = C_u + C_{uv} = C_u + d_{uv} - \pi_u$；若節點 v 為收貨任務點則「已收未送的訂單集合」$PickupSet_v = PickupSet_u \cap v$，若節點 v 為送貨任務點則$PickupSet_v = PickupSet_v \backslash v$。將節點 v 設為已嘗試，到步驟 A-5。

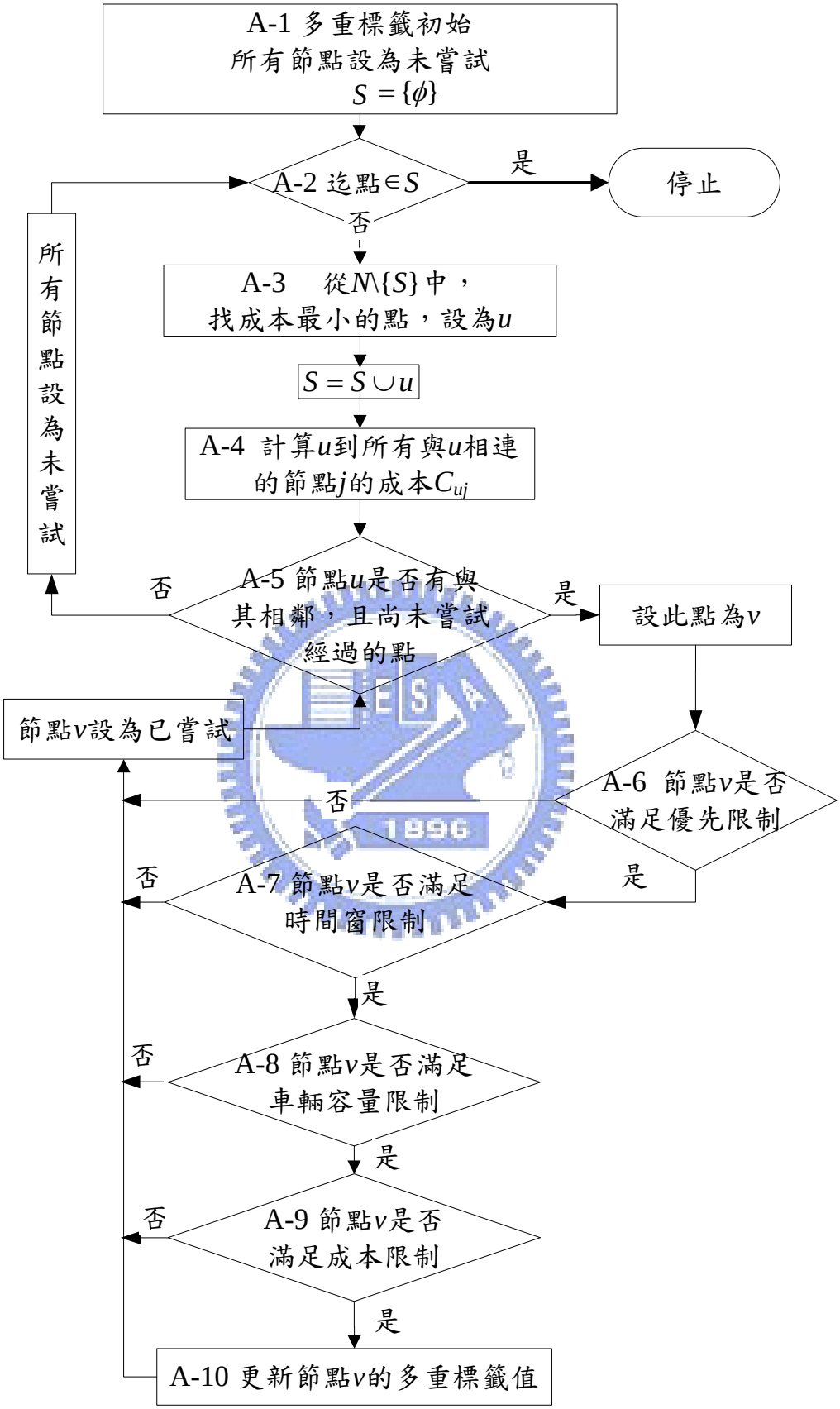
A-1 多重標籤初始
所有節點設為未嘗試
$S=\{\phi\}$
A-2 迄點$\in S$
是
停止
否
所有節點設為未嘗試
A-3 從$N\backslash\{S\}$中，
找成本最小的點，設為u
$S=S\cup u$
A-4 計算u到所有與u相連
的節點j的成本C_{uj}
A-5 節點u是否有與
其相鄰，且尚未嘗試
經過的點
否
是
設此點為v
節點v設為已嘗試
A-6 節點v是否
滿足優先限制
否
是
否
A-7 節點v是否滿足
時間窗限制
是
否
A-8 節點v是否滿足
車輛容量限制
是
否
A-9 節點v是否
滿足成本限制
是
A-10 更新節點v的多重標籤值

圖 9 演算法 A 之流程圖

(三) 演算法 B

演算法 A 為修正傳統 Dijkstra's Algorithm 而得，但演算法 A 中隱含了無法求得一條從起點至迄點的可行路徑之缺點。本小節將詳述演算法 A 之缺點、改善演算法 A，增加演算法 B 之方法與演算法 B 之演算流程。

1. 演算法 A 之缺點

由於演算法 A 除了成本限制外尚包含收送貨問題特性之四種限制，使得其中某些路徑必須執行某些特定節點才可回到場站(聯結限制、收送貨合理性)形成一條路徑。即使迄點尚未有上游點，在其他節點均已經被固定住之情形下，迄點仍會被固定而結束演算法。詳細說明如下。

圖 10 之表示方式與圖 6 之定義相同。即節點內數字為節點編號，以黑色節點表示起迄點；白色節點表示收貨點；灰色節點表示送貨點。編號 0 與節點編號 7 分別為起迄點；節點編號 1~3 為收貨任務點，需求載重量為正值；節點編號 4~6 為送貨任務點，需求載重量為負值。節點編號 1、4 為同一訂單；節點編號 2、5 為同一訂單；同理，節點編號 3、6 為同一訂單。

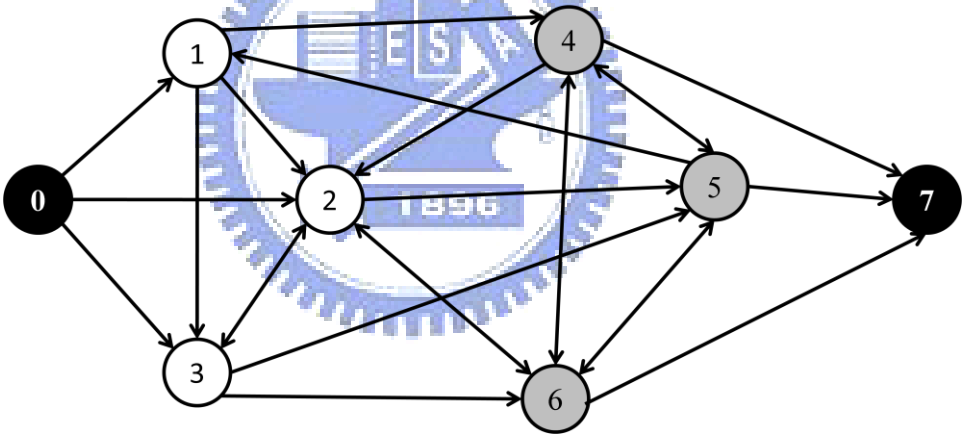

圖 10 演算法 A 之缺點說明一

假設演算法 A 已執行一部分，目前找到之部分路徑為 0→1→2→3→6，即該路線已涵蓋三筆訂單之收貨服務，並已服務第三筆訂單之送貨服務。故於演算法 A 中，節點 0、1、2 與 3 之標籤已被設為固定{0,1,2,3}∈S，如圖 11 所示，圖中*為標籤已被設為固定之節點，箭頭為車輛行駛路徑。

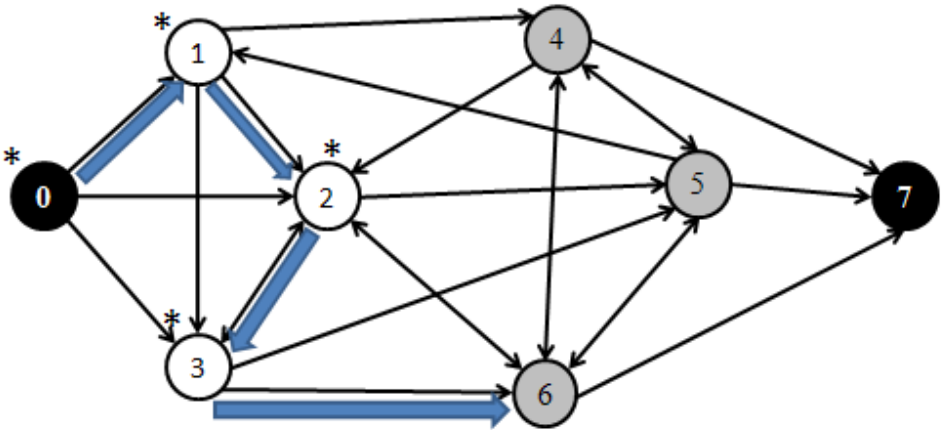

圖 11 演算法 A 之缺點說明二

演算法 A 繼續執行，標籤尚未被設成固定的節點中，節點 6 之成本最小，故節點 6 之標籤被設成固定。此時與節點 6 相鄰且標籤尚未被設成固定的節點有節點 4、節點 5 與迄點 7。因此時車上尚有第一筆與第二筆訂單之貨物，故不允許回迄點 7；假設從節點 6 至節點 4 與節點 5 皆滿足優先、時間窗、車容量、成本等限制，則可更新節點 4 與節點 5 之標籤資訊。如圖 12 所示。

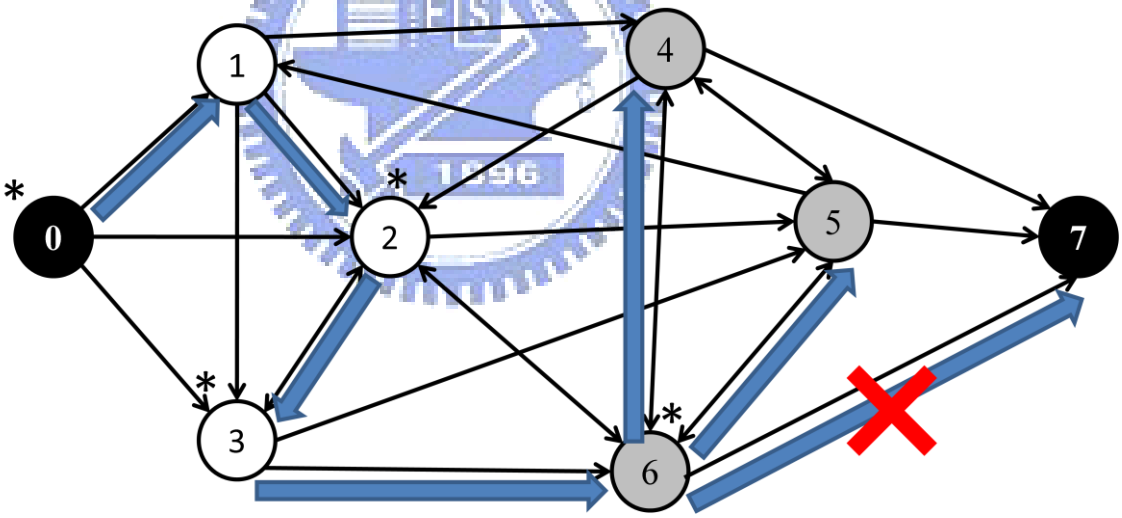

圖 12 演算法 A 之缺點說明三

演算法 A 繼續執行，假設標籤尚未被設成固定的節點中，節點 5 之成本最小，故將節點 5 之標籤設為固定。此時與節點 5 相鄰且標籤尚未被設成固定之節點有節點 4 與迄點 7。因此時車上尚有第一筆訂單之貨物，故不允許回迄點 7；又假設從節點 5 至節點 4 不滿足時間窗限制，故於此迭代中，僅固定節點 5 之標籤，未更新任何節點。如圖 13 所示。

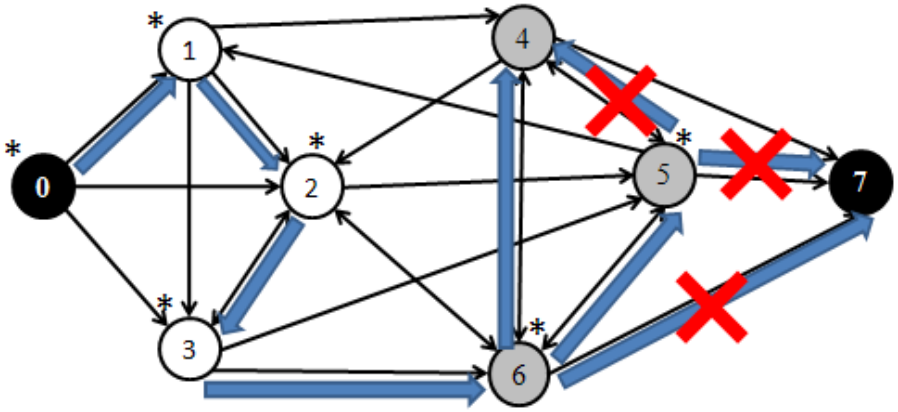

圖 13 演算法 A 之缺點說明四

演算法 A 繼續執行，標籤尚未被設成固定的節點中，僅剩節點 4 與迄點 7，節點 4 之成本最小，故將節點 4 之標籤設為固定。此時與節點 4 相鄰且標籤尚未被設成固定的節點僅剩迄點 7。由於此時車上尚有第二筆訂單之貨物，故不允許回迄點 7。如圖 14 所示。

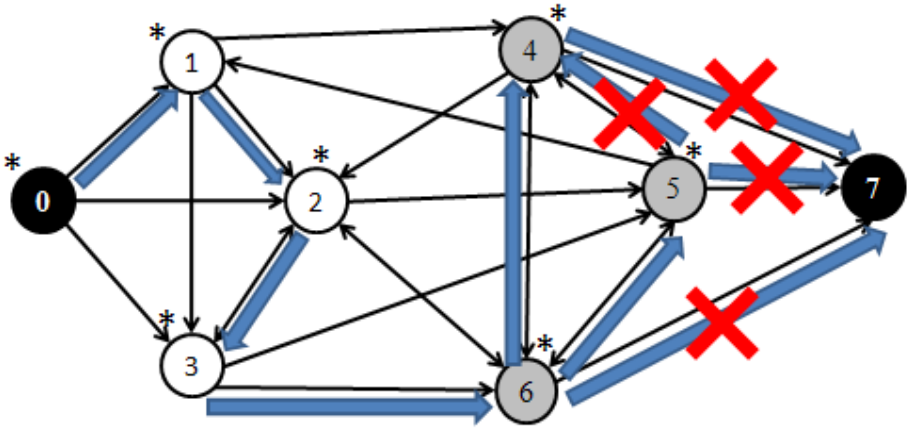

圖 14 演算法 A 之缺點說明五

演算法 A 繼續執行，除了迄點 7 以外之所有節點的標籤皆已被設成固定，故此時將固定迄點 7 之標籤，而結束演算法，但此時迄點 7 無上游點，無法得到一條從起點 0 至迄點 7 之完整可行路徑。

2. 改善演算法 A 之缺點---修補路線與改善路徑成本值

本研究為改善演算法 A 之缺點，設計演算法 B，利用最省插入法修補路線，產生一條完整路徑，並使用交換法改善修補完整之路徑降低成本值，以期產生一條負成本的路徑。故演算法 B 又可分為兩演算法：B1 為修補路線演算法，用於演算法 A 執行完畢後，迄點無上游點時；B2 為改善成本演算法，可用於路線修補完整後與演算法 A 執行完畢後且迄點有上游點時。

3. 路線修補 B1 演算法之執行步驟

令 D 為送貨點已插入或收貨點已移除的訂單集合，U 為送貨點已嘗試插入但未插入或收貨點已嘗試移除但未移除的訂單集合。演算法 B1 步驟如下，流程圖如圖 15 所示。

步驟B1-1. 找成本最小的未完成路徑，到步驟 B1-2。

步驟B1-2. 判斷該路徑「已收未送的訂單集合」中是否有不包含於 D 和 U 內之訂單，若有，選取一在「已收未送的訂單集合」中且不含於 D 和 U 內之訂單的送貨點設為 d，到步驟 B1-3。反之，到步驟 B1-5。

步驟B1-3. (最省插入)在使整條路徑之所有節點皆滿足優先順序、時間窗限制、與車容量限制之條件下，將節點 d 依最省插入法插入該路線。若節點 d 可插入，將 d 點所屬之訂單放入集合 D 中，回步驟 B1-2。若節點 d 不可插入，到步驟 B1-4。

步驟B1-4. (移除)在使整條路徑之所有節點皆滿足優先順序、時間窗限制、與車容量限制之條件下，將與節點 d 配對之收貨點從路徑中移除。若該收貨點可移除，將 d 點所屬之訂單放入集合 D 中；若該收貨點不可移除，將節點 d 所屬之訂單放入集合 U 中。回步驟 B1-2。

步驟B1-5. 判斷集合 U 中是否有訂單，若有，將集合 U 清空，到步驟 B1-2 。若無，使路徑回到迄點，演算法停止。

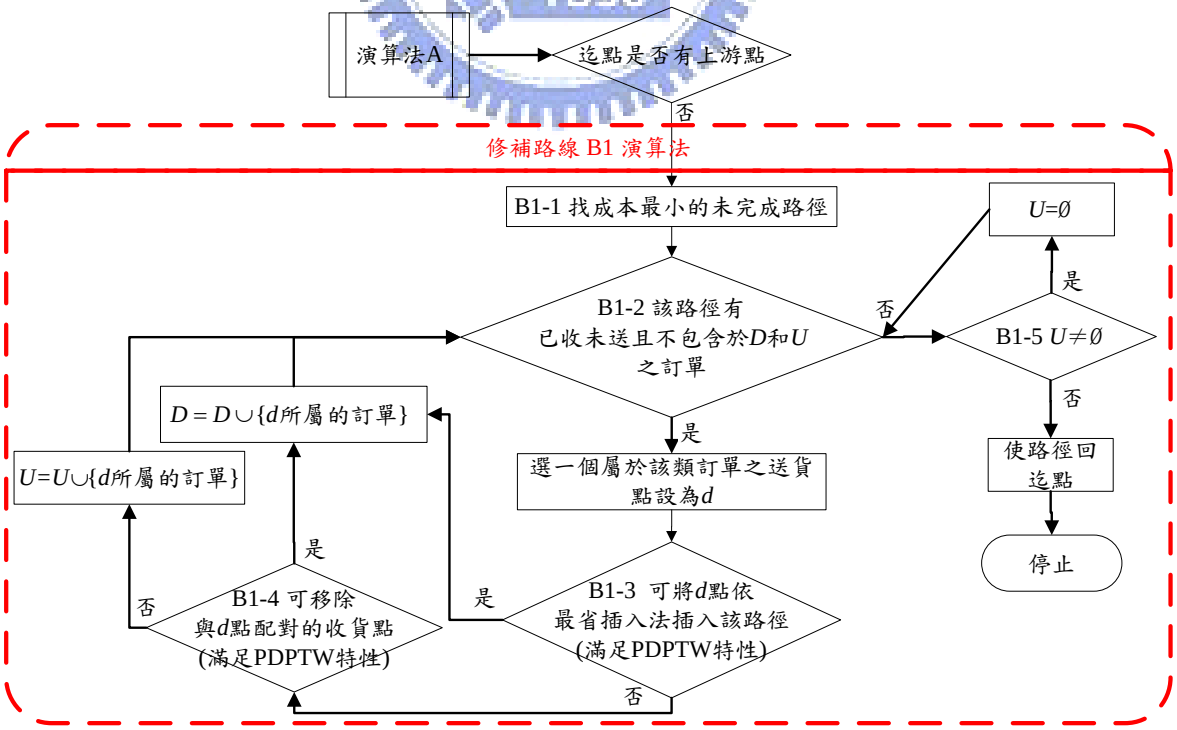

圖 15 修補路線演算法 B1 之流程圖

4. 路線修補 B1 演算法之範例

本研究以圖 14 迄點沒有上游點為例，執行 B1 演算法。

B1-1 假設目前成本最小的未完成路徑為 0→1→2→3→6→5。如圖 16 所示。

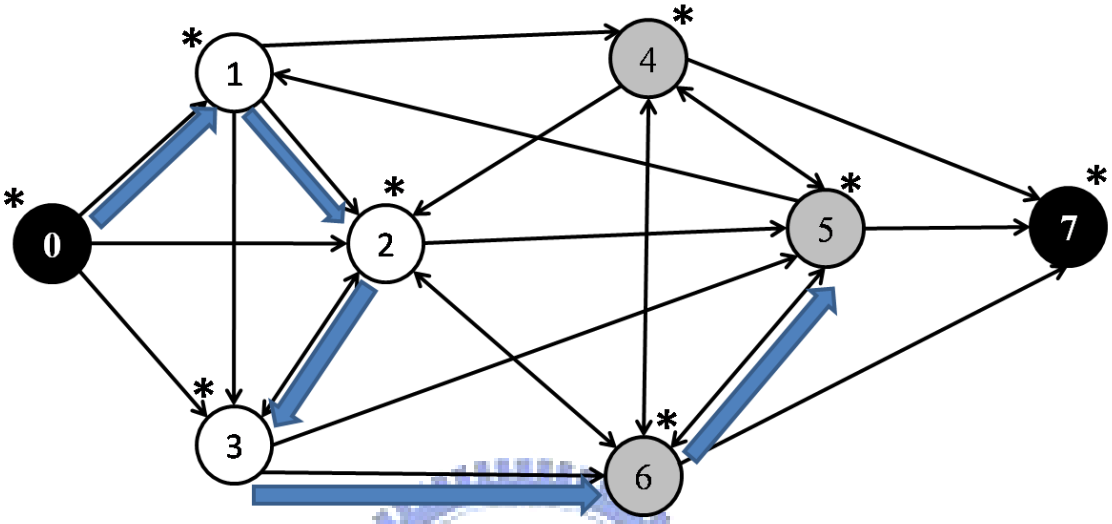

圖 16 路線修補 B1 演算法之範例說明一

B1-2 該路線的「已收未送的訂單集合」={1}，訂單一之送貨點尚未嘗試插入或刪除，故選擇送貨點 d=4。

B1-3 將節點 4 插入 0→1→2→3→6→5→7 中，插入後之路徑排列可能性有多種，選擇滿足時間窗、優先等限制與插入後成本最小者。假設插入後得到之路徑為 0→1→4→2→3→6→5，如圖 17 所示。D={4}。

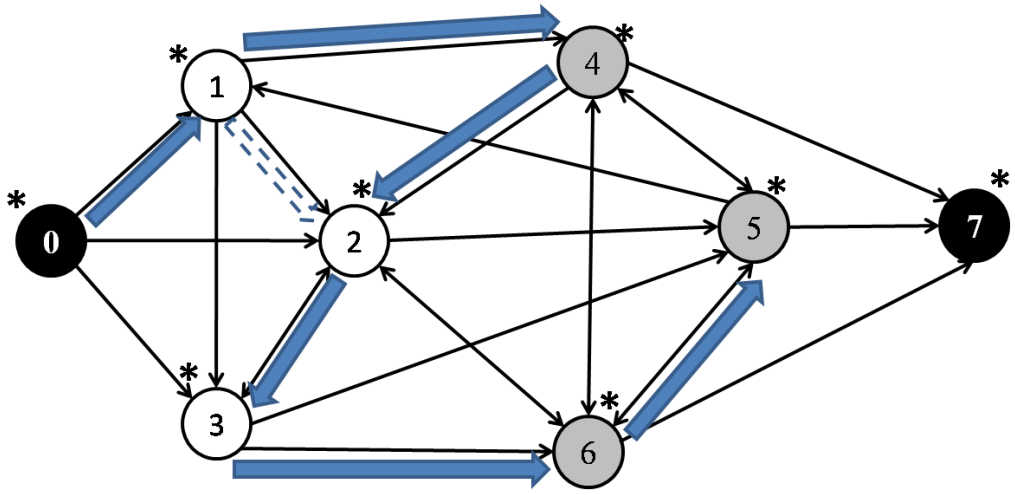

圖 17 路線修補 B1 演算法之範例說明二

B1-2 該路線的「已收未送的訂單集合」={1}，且訂單一之送貨點已嘗試插入。

B1-5 因 $U=\{\emptyset\}$，使路徑回迄點，B1 演算法停止。

5. 成本改善 B2 演算法之執行步驟

演算法 B2 步驟如下，流程圖如圖 18 所示。

步驟B2-1. 找對偶值合最大且不屬於該路徑的訂單需求(收送貨配對)，設訂單需求之收貨點為 e、送貨點為 e'。

步驟B2-2. 在使整條路徑之所有節點皆滿足優先順序、時間窗限制、與車容量限制之條件下，判斷 e 和 e'可否依最省插入法插入該路線，且插入後的路徑成本比未插入的路徑成本小。若可插入且成本較小，到步驟 B2-3；若不可插入或可插入但成本未較小，則保持原路線, 到步驟 B2-4。

步驟B2-3. 則將 e,e'插入路線，並將所有點視為尚未嘗試交換，使曾經嘗試交換之節點因新路徑而有機會再嘗試交換，到步驟 B2-4。

步驟B2-4. 找路徑中對偶值合最小的收送貨配對，設為 d 和 d' 。

步驟B2-5. 在使整條路徑之所有節點皆滿足優先順序、時間窗限制、與車容量限制之條件下，判斷 d 和 d'可否從路線中移除，且移除後的路徑成本比未插入的路徑成本小。若可移除且成本較小，則將 d 和 d'從路線中移除，到步驟 B2-6；若不可移除或可移除但成本未較小，則保持原路線，到步驟 B2-7。

步驟B2-6. 判斷路徑成本是否小於零。若小於零，則演算法停止；若不小於零，則到步驟 B2-7。

步驟B2-7. 判斷是否尚有未嘗試交換的點。若是，則回到步驟 B2-1；若否，則演算法停止。

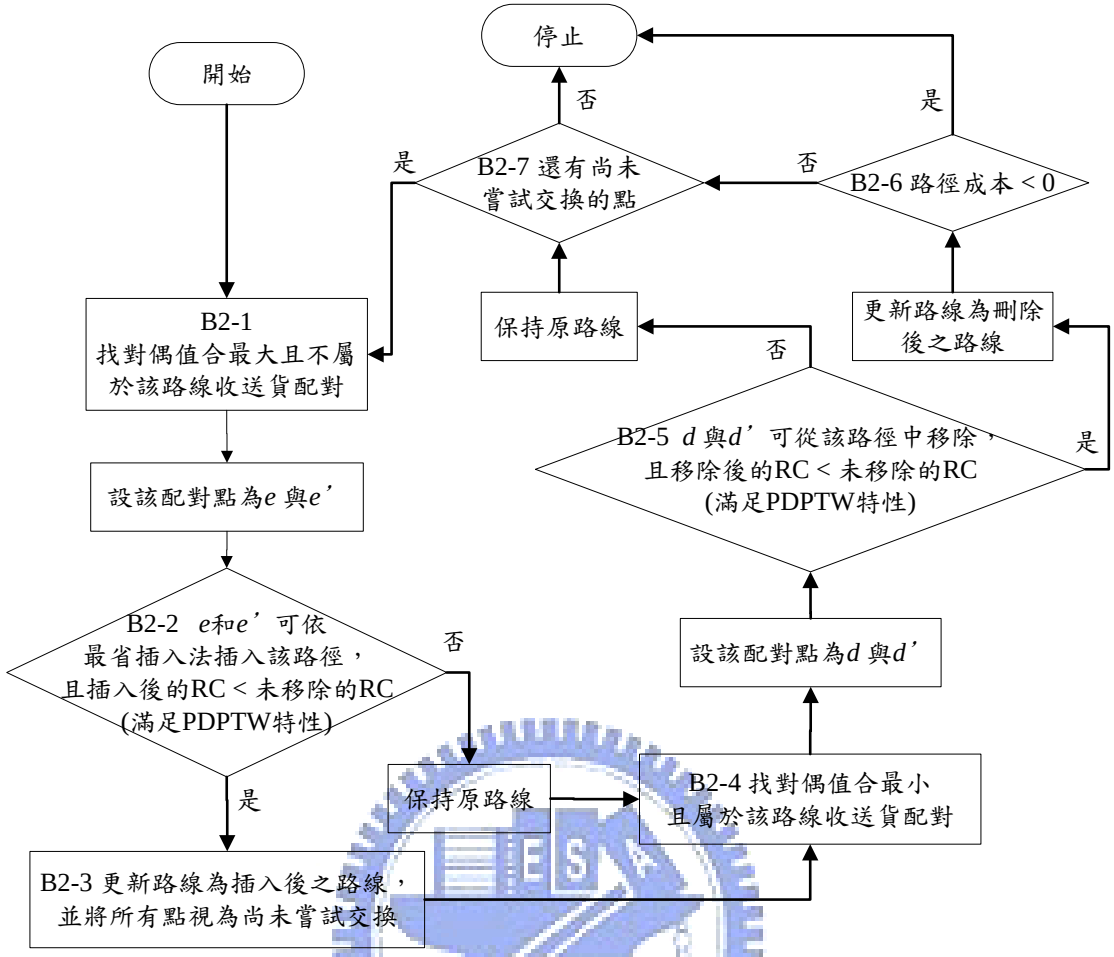

圖 18 改善成本演算法 B2 之流程圖

6. 成本改善 B2 演算法之演算範例

本研究以圖 10 路網為例，執行 B2 演算法。假設目前之路徑為 0→1→2→4→5→7。如圖 19 所示。

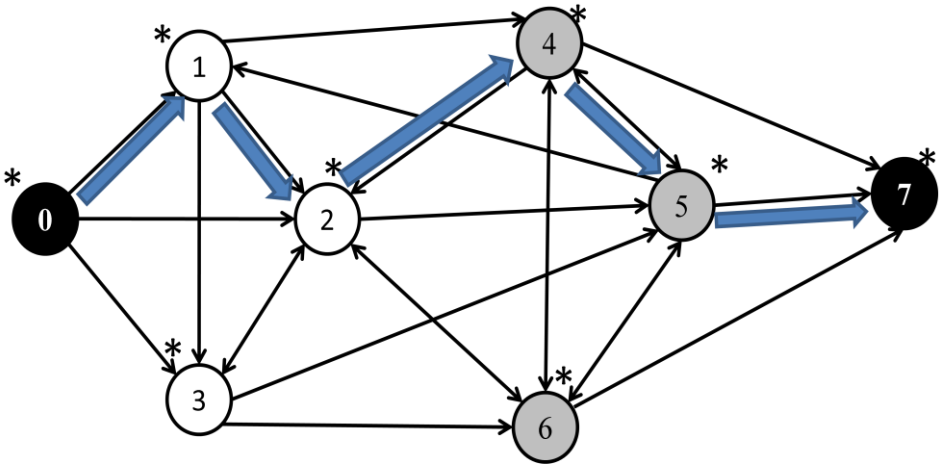

圖 19 成本改善 B2 演算法之範例說明一

B2-1 對偶值合最大，且不屬於該路線的訂單為第三筆訂單。即節點 3 與節點 6。

B2-2 假設在滿足優先順序、時間窗限制、車容量限制之條件下，第三筆訂單之收送貨點可插入目前路徑(有多種插入方式，選擇插入後成本最小者)，且插入後之路徑成本比原路徑成本小。

B2-3 將第三筆訂單插入目前路徑，假設為 0→1→3→2→4→6→5→7。將所有點視為尚未嘗試插入與移除。(使其未來可重複嘗試)

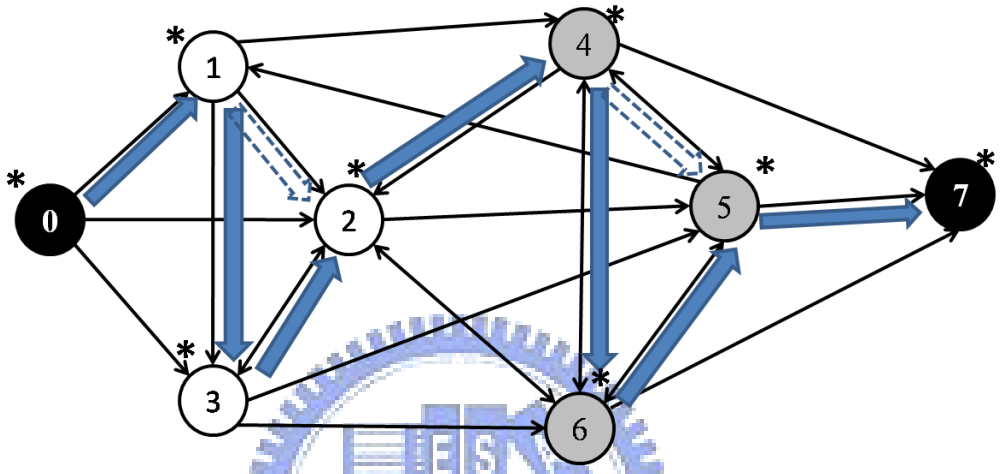

圖 20 成本改善 B2 演算法之範例說明二

B2-4 找目前路徑中對偶值合最小的收送貨配對者，假設為第一筆訂單，即節點 1 和節點 4。

B2-5 假設在滿足優先順序、時間窗限制、車容量限制之條件下，第一筆訂單之收送貨點可從目前路徑中移除，且移除後路徑成本比原路徑成本小。故移除節點 1 和節點 4，並將其標示為已嘗試移除。路徑為 0→3→2→6→5→7。

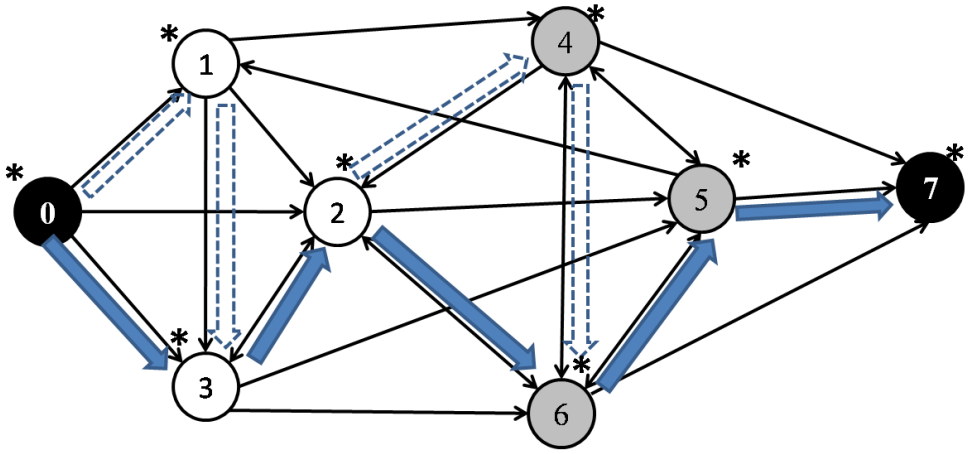

圖 21 成本改善 B2 演算法之範例說明三

B2-6 假設目前路徑成本為負，演算法停止。

3.4 處理重複涵蓋之啟發式演算法

本研究利用移除法修正由變數產生法求解所產生重複涵蓋之情形，即單一任務被兩台車以上所服務。本演算法主要觀念為逐一找出重覆涵蓋節點的路徑，計算移除該重覆涵蓋節點後之可節省之路徑成本，留下節省路徑成本最少的節點，剩餘節點皆刪除，使該節點僅被一輛車服務。流程圖如圖 22 所示。

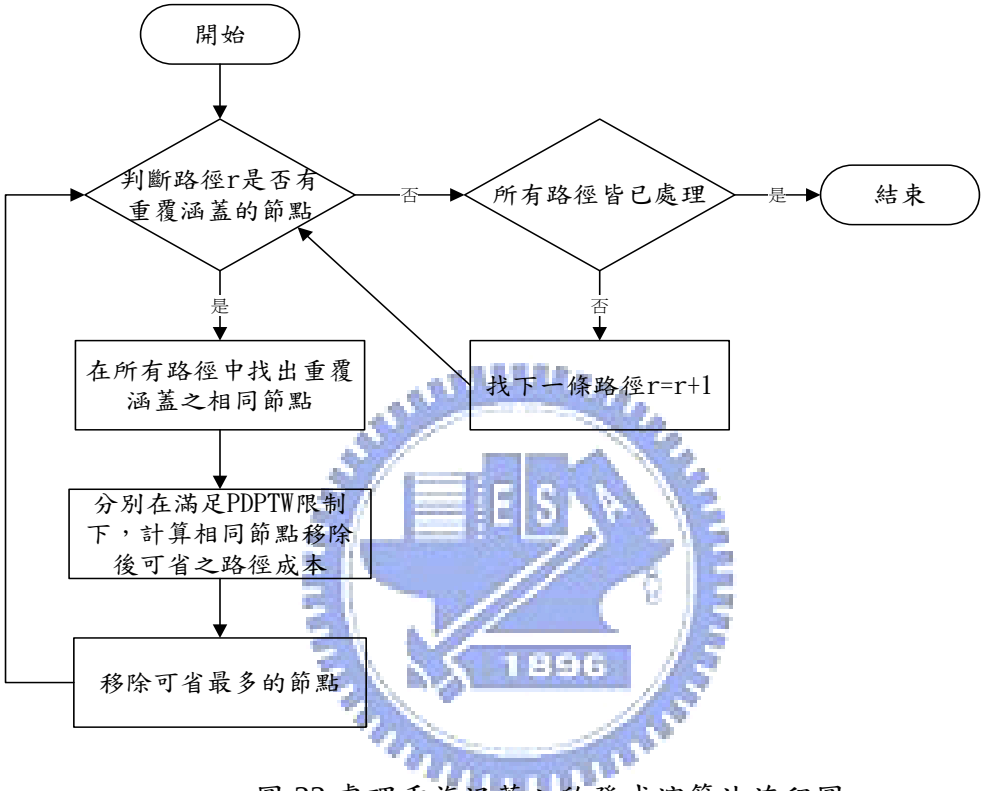

圖 22 處理重複涵蓋之啟發式演算法流程圖

第四章　近似最佳解演算法

為了解本研究所建構之演算法 DECAP 與最佳解之差距，本研究再建構一近似最佳解演算法，稱為演算法 OPT。演算法 OPT 亦使用變數產生法，演算法 OPT 流程架構與演算法 DECAP 相同，如圖 4 所示，僅於求解子問題之演算法不同，故本章節將說明演算法 OPT 之子問題演算法。

4.1 求解子問題演算法概念

本研究為了確保最佳性，將最佳解演算法之子問題以標籤修正法(Label Correcting Method)為基礎，求得對受限主問題有用之路線，加入受限主問題中求解。傳統標籤修正法於每次迭代時，依序從一記錄容器內取出一節點當作目前終點，更新目前終點之下游節點上的成本標籤。若下游點之成本標籤值比原成本還小時，則將其成本標籤與前置點更新，並將該節點編號放入記錄容器，做為下次迭代的目前終點。

如 0 節所述，本研究之子問題為有限制之最短路徑問題，故最佳解演算法之子問題亦應用多重標籤觀念，修正標籤修正法為多重標籤修正法。多重標籤與本研究所建構之演算法相同，包含「成本 C_i」、「前置點 P_i」、「執行時間 T_i」、「累積載重 L_i」，與「已收未送的訂單集合 $PickupSet_i$」。多重標籤修正法更新節點之檢驗條件與 0 節所述相同，但傳統標籤修正法之成本限制會導致無法求出某些路徑，為確保最佳性，故不檢驗成本限制，如 4.3 節所述。

4.2 示意圖說明

為解釋多重標籤修正法每迭代所更新的節點，本研究將路網之每迭代更新之節點依迭代順序呈現，即將網路圖轉為樹狀圖呈現，如圖 23 與圖 24 所示。圖 23 與圖 24 之括號內僅標示出(「成本」,「已收未送的訂單集合」)兩標籤，迭代 1 為由節點 0 更新節點 1、節點 2 與節點 3，迭代 2 為由節點 1、節點 2 與節點 3 分別更新其他點，例如由節點 1 更新節點 2 和節點 4。

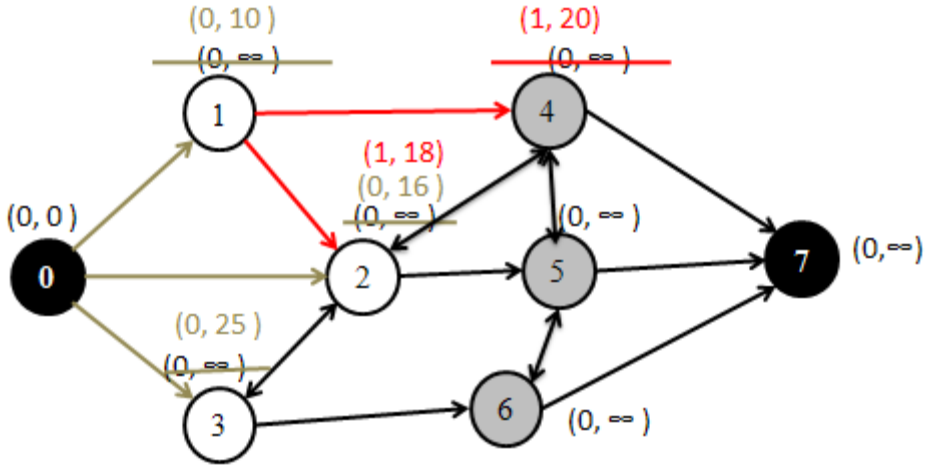

圖 23 網路圖

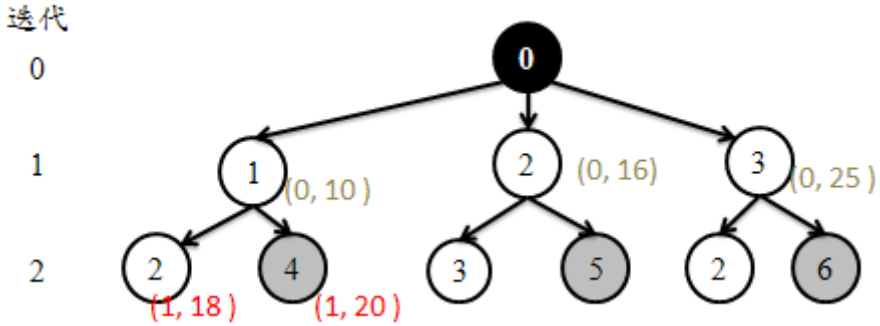

圖 24 由圖 23 轉成之樹狀圖

4.3 更新條件包含成本之缺點

範例OPT-1：

以一含有 8 個節點(3 筆訂單)的收送貨路網為例，節點 1、2 和 3 為收貨點；節點 4、5 和 6 為送貨點；節點 0 和節點 9 為起迄點。起迄點以黑色節點表示；收貨點以白色節點表示；送貨點以灰色節點表示。假設時間窗與車輛容量限制在迭代 4 之前皆符合，且前置點可直接由圖中看出，故括號內僅標示出(「成本」,「已收未送的訂單集合」)兩標籤。

由圖 25 可知，迭代 3 節點編號 6 之成本為-79，「已收未送的訂單集合」為 2，表示經過節點 6 後必執行節點編號 5(2+3=5)才可回迄點；而由節點編號 6 直接執行節點編號 5 的成本為-38，比由迭代 3 節點編號 1 直接執行節點編號 5 的成本-61 還大，故在迭代 4 前置點為 6 的節點編號 5 不會被更新，則此條路徑即使最終為最短路徑亦無法被找到。

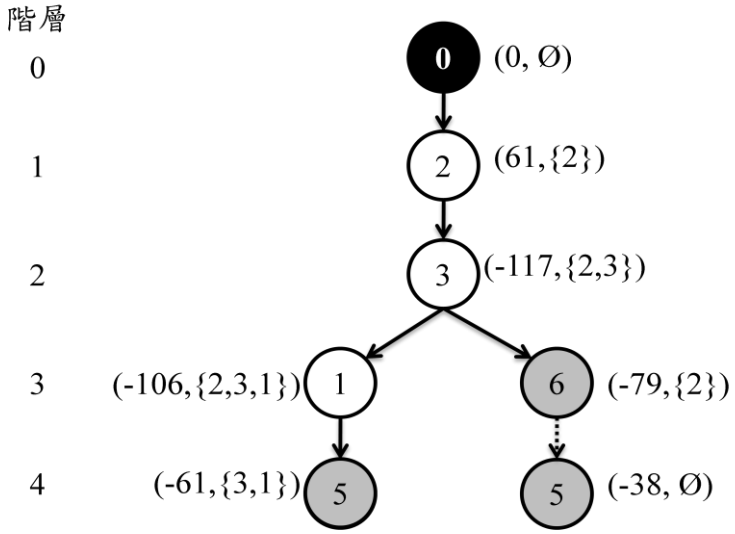

圖 25 檢驗條件包含成本之範例 OPT-1

若不檢驗成本限制，即不論成本是否較小，於迭代 4 前置點為 6 的節點編號 5 將可被更新，如圖 26，可發現其可形成一條成本為負的路徑；而迭代 4 前置點為 1 的節點編號 5 需再經過節點 4 和節點 6 才可回迄點，但由節點編號 5 走至節點 6 時即超過節點 6 之最晚可執行時間，因此此條路徑即使先走節點 4 亦無法完成。

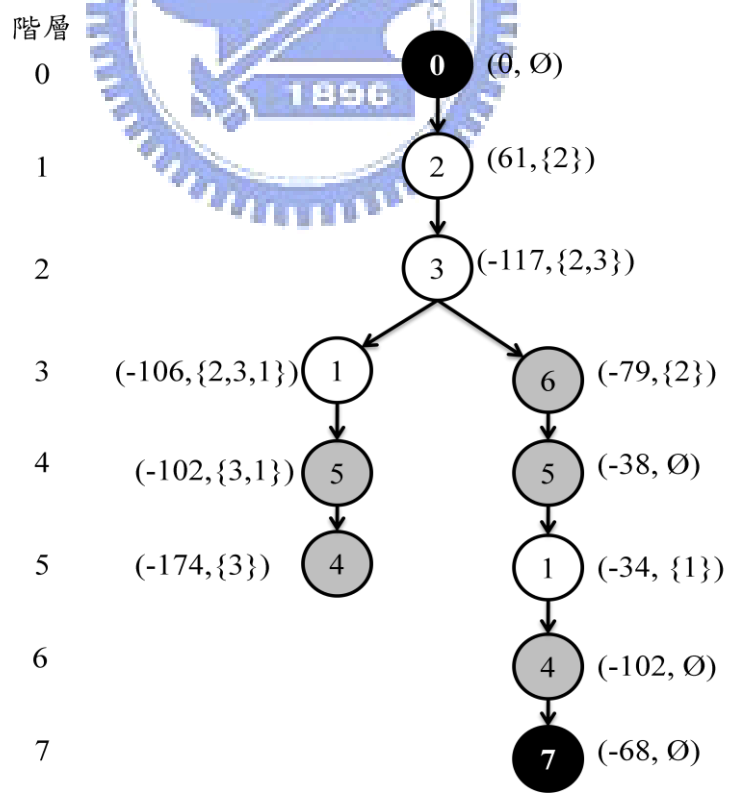

圖 26 檢驗條件不包含成本之範例 OPT-1

由圖 25 圖 26 可看出，檢驗條件包含成本的多重標籤演算法，某些路徑於產生途中會因某些必執行點之成本無法更新而無法形成一條完整路徑；若其成本可被更新，則有機會形成一條有負成本的路徑。

4.4 近似最佳解演算法之子問題執行步驟

本研究為確保最佳性，故不考慮成本之限制，即只需符合 0 小節所提之五種限制，而不檢驗成本限制，即可更新多重標籤。不考慮成本限制之演算法於每一迭代所更新的節點會有許多不同路徑但相同的節點，如圖 27 所示，在迭代 2 中可發現，節點 1 可經由兩條不同的路徑抵達，以粗線條節點表示。因此需額外記錄每一節點多重標籤所屬迭代，並記錄同迭代中其為第幾個相同的節點，以底線表示，以區別相同節點但卻屬於不同路徑。

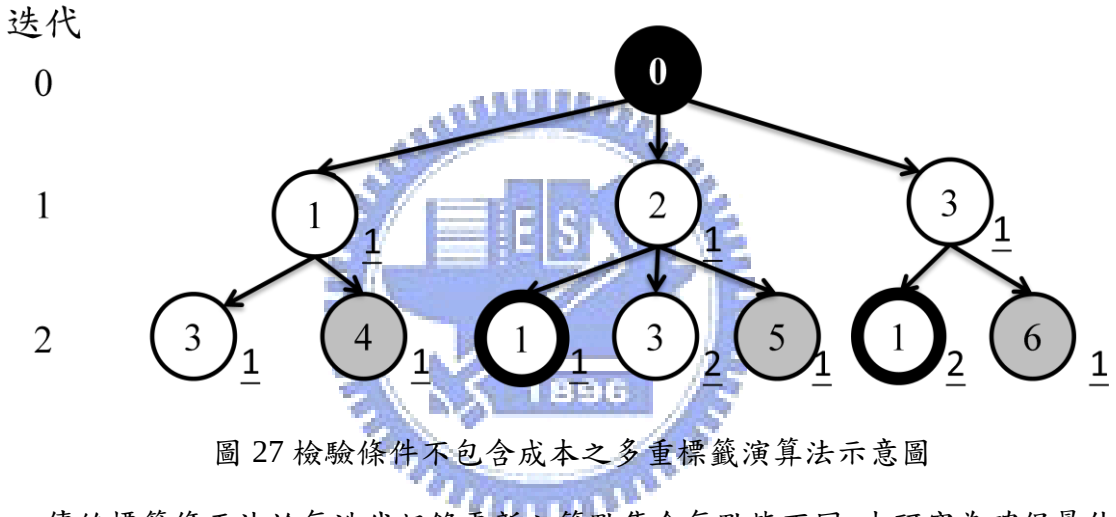

圖 27 檢驗條件不包含成本之多重標籤演算法示意圖

傳統標籤修正法於每迭代記錄更新之節點集合每點皆不同，本研究為確保最佳性，此最佳解演算法每迭代更新之節點則會有相同節點之情形，故需記錄在第 k 迭代中第 i 個節點為第幾個相同的節點之編號。步驟如下，流程圖如圖 28 所示。

令 k 為迭代次數；n 為路網節點數；N 為所有節點之集合；Cap 為車容量上限；$S^{(k)}$為在第 k 迭代所有更新點之集合，其除了記錄更新之節點編號外，尚記錄在第 k 迭代中某節點為第幾個相同的節點之編號，$S^{(k)}$={(更新的節點 i，此節點 i 在第 k 迭代同節點 i 中更新之順序)}。為分辨每一節點於每一迭代是否已試著更新，故將每一點分類為已嘗試或未嘗試。為分辨所有於 $S^{(k)}$中的節點，是否已檢驗其到達所有必執行點之時間在該必執行點之時間窗內，故再區分為已檢驗與未檢驗。

OPT-1. (多重標籤初始化)設定所有點的「前置點」為零、「累積載重」為零、起點的「到達時間」為零，其他點的「到達時間」為無窮大，起點的「累積成本」為零，其他點的「累積成本」為無窮大，所有點的「已收未送的訂單集合」設為空集合。(資料初始化)迭代 k 為 0；$S^{(0)}$={(0,1)}；$S^{(k)}$= ∅，k≠0。

OPT-2. 迭代 k 加 1，將所有屬於 $S^{(k-1)}$的點設為未檢驗。

OPT-3. (停止條件)若迭代 k 小於等於 n-1 且第 k 迭代所有更新點的集合 $S^{(k)}\neq\emptyset$，則到 OPT-4。反之，則演算法停止，找到最短路徑。

OPT-4. 令 u 屬於 $S^{(k-1)}$中的其中一點，對所有與 u 相連的節點 $j\epsilon N$，計算節線成本 $C_{uj}=d_{uj}-\pi_j$。

OPT-5. 令節點 u 之標籤「已收未送的訂單集合」中各訂單之送貨點為 y，對所有節點 y，檢驗由節點 u 到節點 y 的執行時間是否超過節點 y 的最晚可執行時間。將節點 u 設為已檢驗。若到達其中一節點 y 之時間超過該點最晚可執行時間($T_u+t_{uy}>l_y$)，則 u 不可行，執行 OPT-6；反之，u 可行，將所有點設為未嘗試，執行 OPT-7。

OPT-6. 判斷 $S^{(k-1)}$內的點是否皆為已檢驗。若是，則執行 OPT-2；反之，執行 OPT-4。

OPT-7. 判斷節點 u 是否有與其相鄰(u 之下游點)，且未嘗試經過的點。若有，則設該點為 v，並將節點 v 設為已嘗試，到 OPT-8；若無，到 OPT-6。

OPT-8. (優先限制)若節點 v 為收貨點，則滿足優先限制，到 OPT-9。若節點 v 為送貨點，則判斷節點 v 的收貨點是否在節點 u 的「已收未送的訂單集合」中，若在，則滿足優先限制，到 OPT-9；若不在，則違反優先限制，到 OPT-7。

OPT-9. (時間窗限制)若從節點 u 到達節點 v 的時間等於或早於節點 v 最晚可執行時間($T_u+t_{uv}\leq l_v$)，則滿足時間窗限制，到 OPT-10；反之，若從節點 u 到達節點 v 的時間大於節點 v 最晚可執行時間($T_u+t_{uv}>l_v$)，則違反時間窗限制，回到 OPT-7。

OPT-10. (車輛容量限制)若節點 v 為送貨點，則車上總載重減少，必滿足車輛容量限制，OPT-11；若節點 v 為收貨點，車上總載重將增加，故若節點 u 的累積載重與節點 v 的需求載重量加總大於車輛容量($L_u+d_v>Cap$)，則違反車輛容量限制，回到 OPT-7；反之，若加總小於等於車容量($L_u+d_v\leq Cap$)，則滿足車容量限制，到 OPT-11。

OPT-11. (將節點 v 放入 $S^{(k)}$中，並更新節點 v 的多重標籤)更新節點 v 的「前置點」為 P_v=u；節點 v 的「到達時間」為 T_v =$Max(T_u+t_{uv},l_v)$ ；到節點 v 車輛的「累積載重量」為$L_v=L_u+d_v$；到 v 點的「累積成本」為$C_v=C_u+C_{uv}=C_u+d_{uv}-\pi_u$；若節點 v 為收貨任務點則「已收未送的訂單集合」$PickupSet_v=PickupSet_u\cup v$，若節點 v 為送貨任務點則$PickupSet_v=PickupSet_v\backslash v$。到 OPT-4。

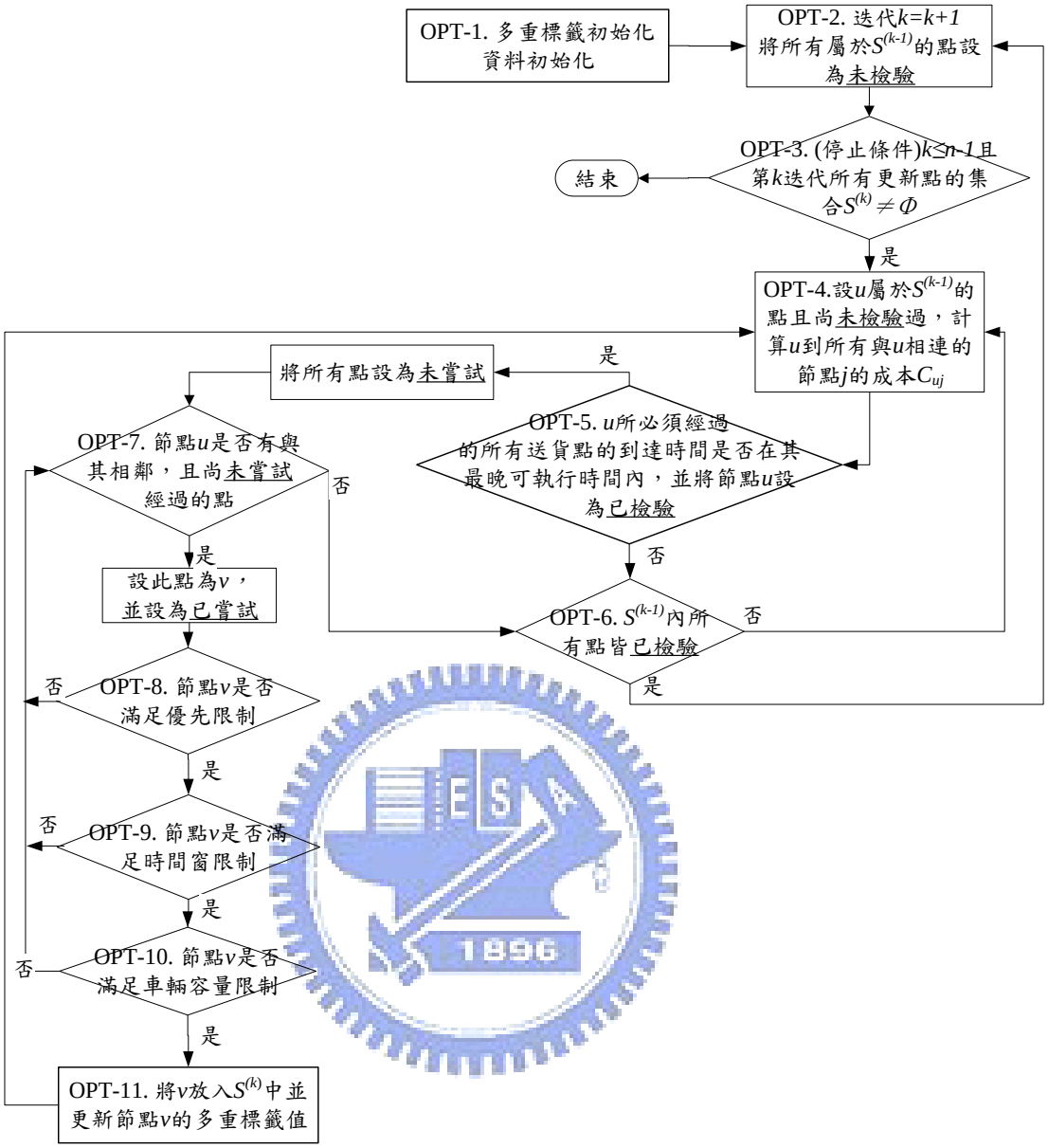

圖 28 最佳解演算法之子問題演算法流程圖

第五章　範例測試

5.1 測試例題

Li and Lim[10]以 Solomon[23]VRPTW 之標竿例題為基礎，產生適用於 PDPTW 的標竿例題[29]。和 SolomonVRPTW 之標竿例題相似，Li and Lim[10]所提出的 PDPTW 標竿例題亦分為六種類型：LC1、LC2、LR1、LR2、LRC1 與 LRC2。依客戶散布型態分成群聚 LC、隨機分配 LR 與一半群聚一半隨機分配 LRC；其中又依時間窗長度分類，LC1、LR1 和 LRC1 的時間窗較短，LC2、LR2 和 LRC2 的時間窗較長。所有例題除了原本 solomon 之客戶點外，為滿足 PDPTW 之特性，尚包含有額外的虛擬客戶點(dummy nodes)。

至今已有許多學者[9][10] [15][18][19]皆依其所欲解決之 PDPTW 特性，依不同方式，將 Solomon VRPTW 之標竿例題轉為不同之 PDPTW 標竿例題。回顧文獻發現，因各學者所產生之 PDPTW 標竿例題僅適用於自己所欲解決之 PDPTW 特性，對於其他學者所欲解決之問題特性不適用；或因欲使用之標竿例題已不復存在，故各學者仍自行產生標竿例題，導致至今尚未有一 PDPTW 標竿例題之影響力與 Solomon 之 VRP 標竿例題相當，即後進學者皆引用該 PDPTW 標竿範例作為演算法之測試比較。故本研究將不與任一標竿範例比較分析，僅使用 Li and Lim[10]所提出之標竿例題作為本研究之路網結構，測試本研究之演算法。

Bell and McMullen[1]於 2004 年使用螞蟻算法求解 VRP，最後使用 ANOVA 分析不同的候選大小與單群聚或多群聚螞蟻演算法對於求解品是否有顯著差異；Saremi et al.[20]於 2007 年使用 Memetic Algorithm 求解 VRPB，最後亦使用 ANOVA 分析不同的交配型態、交配率、突變型態與區域搜尋型態對於求解品質與求解時間是否有顯著差異。

本研究測試分為兩階段，為了解本研究所提出之兩套演算法之求解績效，第一階段利用小規模範例比較本研究所提出之兩套演算法。又為了解本研究演算法 DECAP 所提出改善機制之效用，測試第二階段利用大規模範例，以本研究提出之演算法 DECAP 為基礎產生三種不同演算法，分別對平均旅行成本(目標式)、平均車輛數與平均求解時間進行單變量變異數分析(One Way ANOVA)。

小規模範例網路客戶點數分別包含 6、8、10、16、20、24 與 26 個客戶點。大規模範例以 Li and Lim[10]所產生之 PDPTW 標竿例題之 100、200、400 與 600 個客戶範例為主，分別對六類客戶型態挑選 3 題，亦即每種客戶大小共 18 題，總計 72 題。以本研究演算法 DECAP 為基礎之三種：D1 為本研究所提出之演算法 DECAP；演算法 DECAP 中，由於演算法 B2 之步驟 B2-3 使路徑不斷嘗試插入或刪除，直覺上有可能導致求解時間較長但其求解能力並未相對變好，故演算法 D2 將演算法 B2 之步驟 B2-3

刪除；演算法 DECAP 中，若迄點有上游點時，表已有一條完整路徑，執行演算法 B2 改善路徑成本之必要性亦是本研究欲知道之課題。因此將演算法分為三種演算法詳述如表 6。

本研究使用 Microsoft Visual C++6.0 編譯器以 C++程式語言撰寫演算法，並利用 CPLEX 7.0 所提供之求解功能求解變數產生法中之主問題與求取整數解之分支定限法。並於 2.00GB RAM、Windows XP Professional SP2 作業平台的個人電腦上執行測試。

表 6 演算法分類表

演算法分類	演算法分類描述
演算法 D1	為本研究所提之演算法，其中子問題使用演算法 DECAP，如圖 8。
演算法 D2	為本研究所提之演算法，其中子問題使用演算法 DECAP，但於演算法 B2 之步驟 B2-3 刪除，即只要嘗試插入或刪除之點，當路徑改變時不重複嘗試。
演算法 D3	為本研究所提出之演算法，其中子問題使用演算法 DECAP，但求解子問題時若迄點有上游點時，不論其路徑成本是否小於零，皆不做演算法 B2。

5.2 小規模例題測試---本研究兩套演算法比較

測試第一部分使用小規模例題比較本研究所提之兩套演算法，如表 7 所示。由總旅行距離之誤差可知，本研究演算法 DECAP 於小於 20 個客戶點數的網路規模，與演算法 OPT 所求得之總旅行距離成本相同；客戶點大於 24 個點數的網路規模，於總旅行距離成本雖不相同，但誤差百分比皆在 5%以下。演算法 OPT 與演算法 DECAP 求出之車輛數皆相同，但總求解時間卻從客戶點數 16 開始呈非線性成長。由此分析可知本研究演算法 DECAP 於小於 26 個客戶點數之網路規模可求得與演算法 OPT 相近之近似最佳解，演算法 OPT 雖為近似最佳解演算法，卻無法於有效時間內求得好的解。

表 7 本研究演算法與最佳解之比較

客戶點數	起始解	總旅行距離成本				總求解時間(秒)		總車輛數(輛)	
		演算法		誤差	誤差百分比	演算法		演算法	
		OPT	DECAP			OPT	DECAP	OPT	DECAP
6	197.441	98.8657	98.8657	0	0.00%	0	0	1	1
8	267.092	168.517	168.517	0	0.00%	0	0	2	2
10	333.327	179.752	179.752	0	0.00%	0	0	2	2
16	593.329	314.33	314.33	0	0.00%	20	0	3	3
20	715.386	327.4	327.4	0	0.00%	148	0	3	3
24	803.497	333.37	350.638	17.268	4.92%	3184	0	3	3
26	834.642	391.934	406.899	14.965	3.68%	3226	0	4	4

5.3 大規模例題測試---單變量變異數分析(One Way ANOVA)

本節利用 SPSS 分別對六類型客戶於四種客戶數大小進行平均旅行成本、平均車輛數與平均求解時間之單變量變異數分析(ANOVA) ，流程圖如圖 29 所示。首先使用 Levene Test 做母體變異數齊一之檢定，若不顯著，表示各組樣本之母體變異數為齊一；若顯著，則表示母體變異數不齊一。接著做單變量變異數分析，檢定三種演算法之間是否有顯著差異。當 ANOVA 之檢定結果為顯著，則可做事後檢定(Post Hoc)，檢定哪兩個演算法之間有顯著差異。事後檢定方法依母體變異數是否齊一分別使用不同方法：當各組樣本之母體變異數齊一時，事後檢定用 Tukey 法檢定；當各組樣本之母體變異數為不齊一時，事後檢定使用 Dunnett's C 檢定。

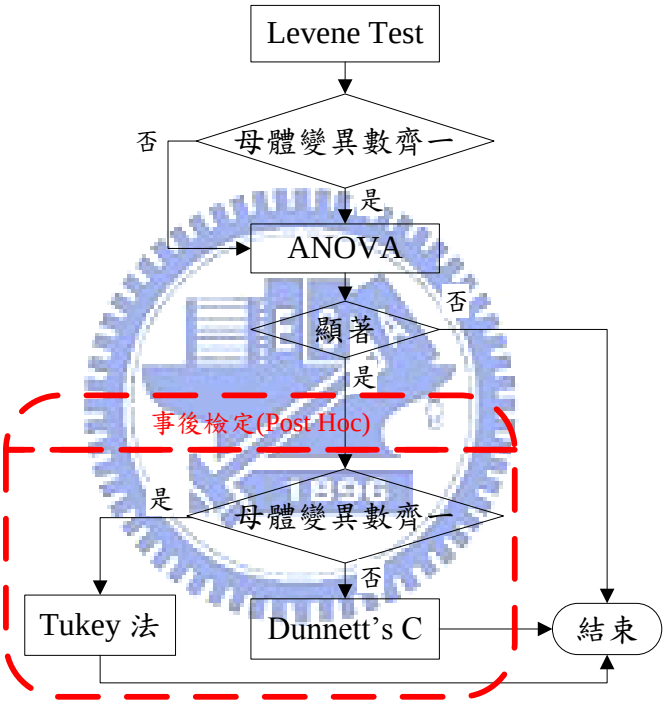

圖 29 ANOVA 流程圖

5.2.1. 平均旅行成本

本小節依六類型客戶與四種客戶數大小，分別使用單變量變異數分析，期望了解各類型與各大小客戶應使用何種演算法可獲得較佳解。

(一) 六類型客戶

平均旅行成本分別對六類型客戶檢定結果如

表 8 所示。由表 8 可知三種演算法應用於時間窗較小之客戶類型有顯著差異。

表 8 平均旅行成本分別對六類型客戶檢定結果

客戶類型	演算法	平均旅行成本	顯著性	事後檢定結果			備註
LC1	D1	4016.3832	0.007*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	6313.3829		演算法 D2	0.871		
	D3	18189.1527		演算法 D3	0.008*	0.025*	
LC2	D1	8183.0497	0.674				
	D2	8120.0700					
	D3	11574.5041					
LR1	D1	9812.7980	0.023*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	9465.3605		演算法 D2	.870		
	D3	20747.5995		演算法 D3	.0*	.0*	
LR2	D1	14857.3168	0.830				
	D2	16138.7150					
	D3	19517.0436					
LRC1	D1	12007.8245	0.047*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	12562.8300		演算法 D2	.960		
	D3	18719.6482		演算法 D3	.018*	.010*	
LRC2	D1	11362.1700	0.048*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	12655.5632		演算法 D2	.821		
	D3	18557.2027		演算法 D3	.002*	.010*	

*表示在水準 0.05 上具有顯著差異。

(二) 客戶數大小

平均旅行成本對四種客戶數大小之檢定結果如表 9。可知三種演算法對客戶數大小為 100、200、400 與 600 皆有顯著差異。

表 9 平均旅行成本分別對六種客戶數大小檢定結果

客戶大小	演算法	平均旅行成本	顯著性	事後檢定結果			備註
100	D1	1562.2353	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	1595.3408		演算法 D2	0.966		
	D3	2524.7862		演算法 D3	0*	0*	
200	D1	5875.5283	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	6057.7688		演算法 D2	0.965		
	D3	9458.1433		演算法 D3	0*	0*	
400	D1	15870.1506	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	18191.1561		演算法 D2	0.536		
	D3	22037.6930		演算法 D3	0*	0*	

600	D1	53594.5000	0.032*	顯著性	演算法 D1	演算法 D2	D2 好於 D3
	D2	50491.6133		演算法 D2	0.9		
	D3	67144.4889		演算法 D3	0.169	0.045*	

*表示在水準 0.05 上具有顯著差異。

(三) 小結

由表 9 可知三種演算法對四種客戶大小皆有顯著效果，且三種演算法對於六種客戶類型，均旅行成本在時間窗小之範例，即 LC1、LR1 及 LRC1，皆有顯著效果，而對於時間窗較大之範例除了混合型態的範例 LRC2 外，無顯著效果。

探究其原因，演算法 D1 與演算法 D2 有較多機會執行演算法 B2 改善成本，演算法 3 除非在迄點無上游點時才有機會執行改善成本演算法 B2。當客戶時間窗範圍較小時，求解過程中容易因時間窗不滿足，僅執行少量訂單，使迄點較易有上游點，則無機會執行演算法 B2 改善目標值；客戶時間窗範圍較大時，求解過程中較容易滿足時間窗限制，可執行訂單收貨之機會多，但收了多數訂單貨之後，回場站前須將收到之貨送完，雖時間窗較大，但所收的貨過多後，亦會不滿足須送貨點之時間窗限制，導致迄點無上游點，而有機會執行演算法 B2。

5.2.2. 平均車輛數

(一) 六類型客戶

平均車輛數分別對六類型客戶檢定結果如表 10 所示。由表 10 可知三種演算法僅對 LR2 客戶類型無顯著差異，但其平均車輛數之差值，亦有明顯之差距。

表 10 平均車輛數分別對六類型客戶檢定結果

客戶類型	演算法	平均車輛數	顯著性	事後檢定結果			備註
LC1	D1	20.2222	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	23.4444		演算法 D2	0.987		
	D3	111.8889		演算法 D3	0.001*	0.001*	
LC2	D1	34.8889	0.041*	顯著性	演算法 D1	演算法 D2	D2 好於 D3
	D2	23.1111		演算法 D2	0.938		
	D3	61.6667		演算法 D3	0.102	0.05*	
LR1	D1	46	0.01*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	40.4444		演算法 D2	0.971		
	D3	112.7778		演算法 D3	0.028*	0.017*	
LR2	D1	405556	0.228				
	D2	37.7778					

	D3	77.6667					
LRC1	D1	31.3333	0.002*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	33.8889		演算法 D2	0.993		
	D3	111.1111		演算法 D3	0.004*	0.005*	
LRC2	D1	27.4444	0.023*	顯著性	演算法 D1	演算法 D2	D2 好於 D3
	D2	21.8889		演算法 D2	0.955		
	D3	73.8889		演算法 D3	0.059	0.032*	

*表示在水準 0.05 上具有顯著差異。

(二) 客戶數大小

平均車輛數分別對四種客戶數大小檢定結果如表 11 所示。由表 11 可知三種演算法對四種客戶數皆有顯著差異。

表 11 平均車輛數分別對六種客戶數大小檢定結果

客戶大小	演算法	平均車輛數	顯著性	事後檢定結果			備註
100	D1	10.9444	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	11.1667		演算法 D2	0.989		
	D3	33.5		演算法 D3	0*	0*	
200	D1	23	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	27.7222		演算法 D2	0.596		
	D3	68.5556		演算法 D3	0*	0*	
400	D1	66.2778	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	51.3889		演算法 D2	0.536		
	D3	172.4444		演算法 D3	0*	0*	
600	D1	168.6364	0*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	165.8667		演算法 D2	0.913		
	D3	273.3333		演算法 D3	0.169	0.045*	

*表示在水準 0.05 上具有顯著差異。

(三) 小結

由表 10 可知小範圍時間窗的客戶類型(LC1、LR1 與 LRC1)，三種演算法對於平均車輛數有顯著效果；大範圍時間窗的客戶類型除了 LR2 未達顯著外，其餘皆達顯著，但 LR2 平均車輛數之差距亦算大，故可推論三種演算法對各種客戶類型皆有明顯的差異效果。

由表 11 可知三種演算法對於平均車輛數於四種客戶數大小之範例有顯著效果，且演算法 D1 之平均車輛數約等於演算法 D2 之平均車輛數，且演算法 D1 和 D2 之平均車輛數少於演算法 D3 之平均車輛數。在事後檢定上，可知演算法 D1 與演算法 D3 有顯著效果，演算法 D2 與演算法 D3 亦有顯著效果，但演算法 D1

與演算法 D2 不顯著。故可推論當客戶數大小相同，可使用演算法 D1 和演算法 D2，以達到較少之車輛數。

歸咎其原因，當演算法於演算法 A 做完時，若迄點有上游點，演算法三皆不會做改善路徑之演算法 B2，故每條路徑之所服務之節點數將較少，所需使用之車輛數則相對較多。

5.2.3. 平均求解時間

本研究於變數產生法後使用分支定限法求得整數解，並非本研究所發展之演算法，故將平均求解時間分為兩種，第一種為包含分之定限法之總求解時間，第二種為不包含分支定限法之總求解時間，由於分支定限法求解時間會依不同範例大小等而有所不同，故本研究使用不包含分支定限法之求解時間作 ANOVA 分析，以具有公平性。

(一) 六類型客戶

平均求解時間分別對六類型客戶檢定結果如表 12 所示。由表 12 可知三種演算法對六類型客戶皆無顯著差異。但由不包含分支定限法之平均求解時間可知，除了 LC2 演算法 D2 之平均求解時間較演算法 D1 久，其餘不包含分支定限法之平均求解時間皆為演算法 D1 大於演算法 D2 大於演算法 D3。

表 12 平均求解時間分別對六種客戶類型檢定結果

客戶類型	演算法	平均求解時間(秒) (包含分支定限法)	平均求解時間(秒) (不包含分支定限法)	顯著性	事後檢定結果
LC1	D1	1747.6667	324.0001	0.259	
	D2	533	130.7870		
	D3	0.8889	0.8889		
LC2	D1	632.6364	628.2855	0.246	
	D2	748.5455	738.7311		
	D3	93.9091	93.8678		
LR1	D1	20.5	19.6297	0.133	
	D2	19.1	15.95		
	D3	1.4	1.4		
LR2	D1	306.5833	305.9739	0.155	
	D2	101.0833	100.7736		
	D3	70.1667	38.8868		
LRC1	D1	89.75	75.0820	0.278	
	D2	126.5833	74.0222		
	D3	3.5	3.4975		
LRC2	D1	288	287.3721	0.069	

	D2	99.0909	97.666		
	D3	22.9091	22.9035		

*表示在水準 0.05 上具有顯著差異。

(二) 客戶數大小

平均求解時間對四種客戶數大小之檢定結果如表 13 所示。由表 13 可知三種演算法對 100 與 200 個客戶點顯著。

表 13 平均求解時間分別對六種客戶數大小檢定結果

客戶大小	演算法	平均求解時間(秒)(包含分支定限法)	平均求解時間(秒)(不包含分支定限法)	顯著性	事後檢定結果			備註
100	D1	22.0677	22.0677	0.016*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	6.8130	6.8130		演算法 D2	0.087		
	D3	1.7752	1.7752		演算法 D3	0.016*	0.756*	
200	D1	112.7222	111.6632	0.017*	顯著性	演算法 D1	演算法 D2	D1 好於 D3 D2 好於 D3
	D2	45.5556	44.4737		演算法 D2	0.112		
	D3	16.5556	16.5435		演算法 D3	0.015*	0.674*	
400	D1	1081.3333	360.4255	0.127				
	D2	737.1111	519.0284					
	D3	53.0556	53.0208					
600	D1	811.3636	805.6718	0.24				
	D2	1940	953.0299					
	D3	58.8333	37.9983					

*表示在水準 0.05 上具有顯著差異。

(三) 小結

由表 12 可知，平均求解時間對六類客戶雖僅有 LR1 顯著，但三種演算法對每一範例之求解時間，皆為演算法 D1 大於演算法 D3，且演算法 D2 大於演算法 D3。由表 13 可知三種演算法對 100 與 200 客戶顯著，對 400 與 600 客戶不顯著，但三種演算法對 400 與 600 之平均求解時間，亦為演算法 D3 最快。

探討其原因，若演算法執行時有執行演算法 B，則會導致較長之求解時間，由於演算法 3 若迄點有上游點則不執行演算法 B，故演算法 3 之求解時間較演算法 D1 與演算法 D2 之求解時間短。

第六章　結論與建議

6.1 結論

本研究目的為針對貨運業之預先派遣作業提供車輛路徑組合，加速貨運業營運效率。為達本研究之目的，本研究提出兩套以最佳解為基礎的演算法求解有時間窗的收送貨問題，演算法 DECAP 與演算法 OPT，此兩套演算法皆包含變數產生法、分支定限法與解決重複涵蓋啟發式解法。兩套演算法之不同處在於變數產生法中之子問題求解演算法，其餘部分皆相同。本研究在變數產生法中將主問題建構為線性規劃問題，子問題則建構為需滿足 PDPTW 限制的最短路徑問題，屬於 NP-Hard 問題，PDPTW 限制包含有時間窗限制、優先限制、聯結限制與車容量限制等，為求解子問題，本研究提出兩種求解子問題之演算法，第一種求解子問題演算法先使用修正後之多重標籤 Dijstra's 演算法(演算法 A)，並提出使目前路徑成為完整路徑之演算法(演算法 B1)與改善成本之演算法(演算法 B2)；第二種求解子問題演算法使用修正後之多重標籤修正法，並將成本限制式移除，使之可求得近似最佳解。

本研究測試範例分為兩部分，第一部分使用小規模範例比較本研究所提出之兩套演算法；第二部分從 Li and Lim [10]的國際標竿範例中，分別從 100、200、400 與 600 個客戶數大小之六類問題中各取三題作為測試例題，並運用變異數分析進行以演算法 DECAP 為基礎之三種演算法(D1、D2 與 D3)之求解比較分析，由測試結果可知：

1　使用小規模範例比較本研究所提出之兩套演算法，演算法 DECAP 應用於小規模路網可求得近似最佳解。

2　本研究所提出之近似最佳解解法演算法 OPT，其雖可求得比演算法 DECAP 更好之解，但當網路規模愈大，求解時間會大增。

3　本研究測試以平均旅行成本為依變數，三種演算法對於小範圍時間窗之客戶群範例有顯著差異，對同一客戶數大小亦有顯著差異。且演算法 D1 較演算法 D3 求解效果好，演算法 D2 較演算法 D3 求解效果好

4　本研究測試以平均車輛數為依變數，三種演算法對於小範圍時間窗之客戶群範例有顯著效果；對於大範圍時間窗之客戶群範例僅 LR2 無顯著差異，但其平均車輛數之差值，亦有明顯之差距。且對於有顯著差異之事後檢定可知演算法 D1 較演算法 D3 平均車輛數少，演算法 D2 較演算法 D3 平均車輛數少。

5　本研究測試以不包含分支定限法之平均求解時間為依變數，三種演算法對於六類型客戶皆無顯著差異。對 100 與 200 客戶數大小有顯著差異。但不論是六類型客戶或同客戶大小，不包含分支定限法之平均求解時間皆為演算法 D3 最快。

6.2 建議

1 經本研究測試後，發現演算法 D1 與演算法 D2 不論於平均旅行成本、平均車輛數與不包含分支定限法之平均求解時間，皆未有顯著差異，但從其平均數可看出演算法 D1 較演算法 D2 之求解效果佳、平均車輛數較少，但求解時間較久。故若不追求更佳之解，可直接應用演算法 D2 即可。

2 本研究分支定限法之求解方式為使用求解 IP 之軟體，故變數愈多求解時間愈長，建議後續研究可自行設計分支策略，提升求解效率。分支策略方法，可對一組目標變數進行分支。選擇節點可使用深度搜尋法、廣度搜尋法或隨機搜尋法。

3 本研究第一套演算法，即演算法 DECAP，於變數產生法之子問題求解方式使用修正後的 Dijkstra's 演算法，由於考慮收送貨特性，會產生迄點無上游點之特性，建議後續持續修正演算法提升修正後的 Dijkstra's 演算法之求解能力。

4 本研究所提出之兩套演算法分別採用修正後的 Dijkstra's 演算法與修正後的 Label Correcting 演算法，由於修正後的 Dijkstra's 演算法仍承襲傳統 Dijkstra's 演算法永久標籤之特性，導致修正後的 Dijkstra's 演算法需再使用啟發式解法才可找到最短路徑；而修正後的 Label Correcting 演算法，亦承襲傳統 Label Correcting 演算法之暫時與永久標籤之特性，故不需再使用啟發式解法即可找到最短路徑。因此建議後續研究採用 Label Correcting 演算法求解子問題網路。

參考文獻

[1] Bell, J. E. and McMullen, P. R., “Ant colony optimization techniques for the vehicle routing problem”, Advanced Engineering Informatics, 18, pp. 41-48, 2004.

[2] Bent, R. and Hentenryck, P.V.,“A Two-stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows” , Computers and Operations Research, Vol. 33, No. 4, pp. 875-893 ,2006.

[3] Berbeglia, G., Cordeau, J., Gribkovskaia, I. and Laporte, G., “Static pickup and delivery problems: a classification scheme and survey”, TOP, Vol 15, No. 1, pp. 1-31, July 2007.

[4] Dantig, G.B. and Wolfe, P., “The Decomposition Algorithm for Linear Programming”, Operations Reserch, Vol. 8, pp.101-111, 1960.

[5] Desaulniers, J. Desrosiers, A. Erdmann, M M. Solomon, and F. Soumis., “VRP with pickup and delivery” , in: P. Toth and D. Vigo (eds.), The Vehicle Routing Problem, volume 9 of SIAM Monographs on Discrete Mathematics and Applications, chapter 9, pp. 225–242. SIAM, Philadelphia, 2002.

[6] Desrochers, M., Lenstra, J. K., Savelsbergh, M.W.P.and Soumis, F., “Vehicle routing with time windows: optimization and approximation” , in: B.L. Golden, A.A. Assad (eds.), Vehicle Routing: Methods and Studies, North-Holland, Amsterdam, pp. 65-84 , 1988.

[7] Dumas, Y., Desrosiers J. and Soumis, F., “The Pickup and Delivery Problem with Time Windows” , European Journal of Operational Research, Vol. 54, No. 1, pp. 7-22, 1991.

[8] Fabián, J. and Pérez, L., “A Meta-heuristic Applied for a Topologic Pickup and Delivery Problem with Time Windows Constraints” , Lecture Notes in Computer Science, Vol. 3516, pp. 924-928, 2005.

[9] Lau, J.C. and Liang, Z., “Pickup and Delivery with time windows: Algorithms and test case generation” , International Journal on Artificial Intelligence Tools, Vol. 11, No. 3, pp. 455-472, 2002.

[10] Li, H. and Lim, A., “A metaheuristic for the pickup and delivery problem with time windows” , in: 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’01). IEEE Computer Society, Los Alamitos, CA, pp. 333-340, 2001.

[11] Lu, Q. and Dessouky, M., “An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem” ,Transportation Science, Vol. 38, No. 4, pp. 503-514, 2004.

[12] Lu, Q, and Dessouky, M., “A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows” ,European Journal of Operational Research,

Vol. 175, pp. 672-687, 2006.

[13] Mitrovic-Minic, S. and Laporte, G., “Waiting Strategies for the Dynamic Pickup and Delivery Problem with Time Windows” , Transportation Research Part B, Vol. 38, No. 7, pp. 635-655,2004.

[14] Mitrović-Minić, S and Laporte, G., “The pickup and delivery problem with time windows and transshipment” , INFOR, Vol. 44, No. 3, pp. 217-227, 2006.

[15] Nanry, W. P. and Barnes, J. W., “Solving the Pickup and Delivery Problem with Time Windows Using Reactive Tabu Search” , Transportation Research Part B, Vol. 34, No. 2, pp. 107-121, 2000

[16] Parragh, S. N., Doerner, K. F. and Hartl, R. F., “A survey on pickup and delivery problems Part I: Transportation between pickup and delivery locations” , Journal für Betriebswirtschaft, Vol 58, No.1, pp. 21-51, April 2008.

[17] Parragh, S. N., Doerner, K. F. and Hartl, R. F., “A survey on pickup and delivery problems Part II: Transportation between pickup and delivery locations” , Journal für Betriebswirtschaft, Vol 58, No.2, pp. 81-117, June 2008.

[18] Ropke, S., Cordeau, J.-F., “Branch-and-cut-and-price for the pickup and delivery problem with time windows” , Transportation Science, 2007.

[19] Ropke, S.and Pisinger, D. “An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows” , Transportation Science, Vol. 40 , pp. 455-472, 2006.

[20] Saremi, A., ElMekkawy, T. Y., and Wang, G. G., “Tuning the Parameters of a Memetic Algorithm to Solve Vehicle Routing Problem with Vackhauls Using Design of Experiments”, International Journal of Operations Researchn, Vol. 4, No.4, pp. 206-219, 2007.

[21] Savelsbergh, M. W. P. and Solomon, M., “The General Pickup and Delivery Problem”, Transportation Science, Vol. 29, No. 1, pp. 17-29, 1995.

[22] Sigurd, M., Pisinger, D. and Sig, M., “Scheduling transportation of live animals to avoid the spread of diseases” , Transportation Science, Vol. 38, No. 2, pp. 197-209, 2004.

[23] Solomon, M., “Algorithms for the vehicle routing and scheduling problems with time window constraints”, Operations Research , Vol. 35, No. 2, pp. 254-265, 1987.

[24] Toth, P. and Vigo, D., The Vehicle Routing Problem, SIAM, Philadelphia, 2002.

[25] Wang, X., “Algorithms and Strategies for Dynamic Carrier Fleet Operations for an Application to Local Trucking Operations,” University of California at Irvine, U.S.A.,

Ph.D.Dissertation, 2000.

[26] Xu, H., Chen, Z. L., Rajagopal, S., and Arunapuram, S., “Solving a practical pickup and delivery problem” , Transportation Science ,Vol. 37, pp. 347-364, 2003.

[27] 盧宗成,「捷運司機員排班問題之研究-以台北捷運為例」，國立交通大學運輸工程與管理學系碩士論文，民國八十九年六月。

[28] 黃信翔,「解決具時間窗限制的提送貨問題」，國立交通大學運輸科技與管理學系碩士論文，民國九十五年六月。

[29] Unpublished Results SINTEF Applied Mathematics-Department of Optimisation,

Technical Report in Progress http://www.top.sintef.no/ (2004)

簡歷

姓　　名：葉珮婷

籍　　貫：新竹市

出生日期：民國 73 年 10 月 18 日

E-mail：yehpei@gmail.com

學歷：

國立交通大學運輸科技與管理學系碩士班

國立交通大學運輸科技與管理學系

國立新竹女子高級中學